МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ И НАУКИ
КАБАРДИНО-БАЛКАРСКОЙ РЕСПУБЛИКИ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«ДЕТСКАЯ АКАДЕМИЯ ТВОРЧЕСТВА «СОЛНЕЧНЫЙ ГОРОД»

СОГЛАСОВАНО
на заседании Методического совета
Протокол от «09» 06 2026 г. № 5

«УТВЕРЖДАЮ»
Заместитель директора – руководитель
ЦППМСП ГБОУ «ДАТ «Солнечный город»
Минпросвещения КБР
_____ А.М.Пшихачева
Приказ от «10» 06 2026 г. № 285

ДОПОЛНИТЕЛЬНАЯ ОБЩЕРАЗВИВАЮЩАЯ ПРОГРАММА «IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»

Направленность программы: техническая
Уровень программы: базовый
Вид программы: модифицированный
Адресат программы: обучающиеся 10-15 лет
Срок освоения программы: 2 года (288 ч.)
Форма обучения: очная
Автор-составитель программы:
Кравченко Алексей Михайлович,
педагог дополнительного образования.

Нальчик, 2026

СОДЕРЖАНИЕ

Раздел 1. КОМПЛЕКС ОСНОВНЫХ ХАРАКТЕРИСТИК ПРОГРАММЫ

Пояснительная записка

Направленность программы: техническая.

Вид программы: модифицированный.

Уровень программы: базовый.

Нормативно-правовая база, на основе которой разработана программа:

1. Федеральный закон от 29.12.2012г. №273-ФЗ «Об образовании в Российской Федерации» (с изменениями и дополнениями).
2. Национальный проект «Образование».
3. Конвенция ООН о правах ребенка.
4. Приоритетный проект от 30.11.2016г. №11 «Доступное дополнительное образование для детей», утвержденный протоколом заседания президиума при Президенте РФ.
5. Приказ Министерства просвещения Российской Федерации от 15.04.2019г. №170 «Об утверждении методики расчёта показателя национального проекта «Образование» «Доля детей в возрасте от 5 до 18 лет, охваченных дополнительным образованием».
6. Распоряжение Правительства России от 31.03.2022г. №678-р «Об утверждении Концепции развития дополнительного образования детей до 2030 года» (с изменениями и дополнениями).
7. Распоряжение Правительства России от 29.05.2015г. №996-р «Об утверждении Стратегии развития воспитания в Российской Федерации до 2025 года».
8. Федеральный закон от 13.07.2020г. №189-ФЗ «О государственном (муниципальном) социальном заказе на оказание государственных (муниципальных) услуг в социальной сфере».
9. Приказ Министерства просвещения Российской Федерации от 03.09.2019г. № 467 «Об утверждении Целевой модели развития региональной системы дополнительного образования детей» (с изменениями и дополнениями).
10. Приказ Минобрнауки России от 27.07.2022г. №629 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам».
11. Письмо Министерства образования и науки Российской Федерации от 18.11.2015г. №09-3242 «О направлении информации» (вместе с «Методическими рекомендациями по проектированию дополнительных общеразвивающих программ (включая разноуровневые программы).
12. Письмо Министерства просвещения Российской Федерации 23.01.2026г.№АБ-254/06«О направлении информации» (вместе с «Методическими рекомендациями по разработке дополнительных общеразвивающих программ, в том числе в части интеграции с учебными предметами «Труд (технология)», «Музыка», «Изобразительное искусство», «Физическая культура».
13. Письмо Министерства просвещения Российской Федерации от 29.09.2023г. №АБ-3935/06 «О направлении информации» (вместе с «Методическими рекомендациями по формированию механизмов обновления содержания, методов и технологий обучения в системе дополнительного образования детей, направленных на повышение качества дополнительного образования детей, в том числе включение компонентов, обеспечивающих формирование функциональной грамотности и компетентностей, связанных с эмоциональным, физическим, интеллектуальным, духовным развитием человека, значимых для вхождения Российской Федерации в число десяти ведущих стран мира по качеству общего образования, для реализации приоритетных направлений научно-технологического и культурного развития страны».

14. Письмо Министерства просвещения Российской Федерации от 31.01.2022г. №ДГ-245/06 «О направлении методических рекомендаций по реализации дополнительных общеобразовательных программ с применением электронного обучения и дистанционных образовательных технологий».

15. Письмо Минобрнауки России от 29.03.2016г. №ВК-641/09 «О направлении методических рекомендаций» (вместе с «Методическими рекомендациями по реализации адаптированных дополнительных общеобразовательных программ, способствующих социально-психологической реабилитации, профессиональному самоопределению детей с ограниченными возможностями здоровья, включая детей-инвалидов, с учётом их особых образовательных потребностей»).

16. Протокол заочного заседания Рабочей группы по дополнительному образованию детей Экспертного совета Министерства просвещения Российской Федерации по вопросам дополнительного образования детей и взрослых, воспитания и детского отдыха от 22.03.2023г. №Д06-23/06пр.

17. Постановление Главного государственного санитарного врача от 28.09.2020г. №28 «Об утверждении санитарных правил СП 2.4.3648-20 «Санитарно-эпидемиологические требования к организациям воспитания и обучения, отдыха и оздоровления детей и молодежи» (действует до 01.01.2027г.).

18. Постановление Главного государственного санитарного врача от 28.01.2021г. №2 «Об утверждении санитарных правил и норм СП 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» (действует до 01.03.2027г.).

19. Приказ Министерства здравоохранения и социального развития РФ от 26.08.2010г.№761н «Об утверждении Единого квалификационного справочника должностей руководителей, специалистов и служащих, раздел «Квалификационные характеристики должностей работников образования».

20. Приказ Министерства труда и социальной защиты Российской Федерации от 22.09.2021г.№652н«Об утверждении профессионального стандарта «Педагог дополнительного образования детей и взрослых».

21. Приказ Министерства просвещения Российской Федерации от 04.04.2025г. №269 «О продолжительности рабочего времени (нормах часов педагогической работы за ставку заработной платы) педагогических работников организаций, осуществляющих образовательную деятельность по основным и дополнительным общеобразовательным программам, образовательным программам среднего профессионального образования и соответствующим дополнительным профессиональным программам, основным программам профессионального обучения, и о Порядке определения учебной нагрузки указанных педагогических работников, оговариваемой в трудовом договоре, основаниях ее изменения и случаях установления верхнего предела указанной учебной нагрузки».

22. Распоряжение Правительства России от 28.04.2023г. №1105-р «Об утверждении Концепции информационной безопасности детей в Российской Федерации».

23. Приказ Минобрнауки России и Минпросвещения России от 05.08.2020г. №882/391 «Об организации и осуществлении образовательной деятельности при сетевой форме реализации образовательных программ».

24. Письмо Минобрнауки России от 03.04.2015г. №АП-512/02 «О направлении методических рекомендаций по НОКО» (вместе с «Методическими рекомендациями по независимой оценке качества образования образовательной деятельности организаций, осуществляющих образовательную деятельность»).

25. Письмо Минобрнауки России от 28.04.2017г. №ВК-1232/09 «О направлении методических рекомендаций» (вместе с «Методическими рекомендациями по организации независимой оценки качества дополнительного образования детей»).

26. Постановление Правительства России от 20.10.2021г. №1802 «Об утверждении Правил размещения на официальном сайте образовательной организации в информационно-

телекоммуникационной сети «Интернет» и обновления информации об образовательной организации».

27. Приказ Федеральной службы по надзору в сфере образования и науки от 30.04.2026г. №920 «О внесении изменений в Требования к структуре официального сайта образовательной организации в информационно-телекоммуникационной сети «Интернет» и формату представления информации» (вступает в силу с 01.09.2026 г. и действует до 01.03.2028г.).

28. Закон Кабардино-Балкарской Республики от 24.04.2014г. №23-РЗ «Об образовании» (с изменениями и дополнениями).

29. Постановление Правительства КБР от 22.04.2020г. №85-ПП «О межведомственном совете по внедрению и реализации в Кабардино-Балкарской Республике целевой модели развития региональной системы дополнительного образования детей».

30. Распоряжение Правительства КБР от 26.05.2020г. №242-рп «Об утверждении Концепции внедрения модели персонифицированного дополнительного образования детей в КБР».

31. Приказ Минпросвещения КБР от 01.06.2026г. №22/538 «Об утверждении Административного регламента предоставления государственной услуги «Запись на обучение по дополнительной образовательной программе».

32. Приказ Минпросвещения КБР от 22.08.2025г.№22/783 «Об утверждении Правил персонифицированного учета и персонифицированного финансирования дополнительного образования детей, реализуемого в том числе посредством предоставления детям социальных сертификатов в Кабардино-Балкарской Республике».

33. Приказ Минпросвещения КБР от 26.08.2025г. №22/795 «Об обеспечении независимой оценки качества дополнительных общеобразовательных (общеразвивающих) программ (экспертизы образовательных программ) в рамках системы персонифицированного финансирования».

34. Приказ Минпросвещения КБР от 15.08.2025г.№22/749«Об утверждении Региональных требований к регламентации деятельности государственных образовательных учреждений дополнительного образования детей в Кабардино-Балкарской Республике».

35. Письмо Минпросвещения КБР от 20.06.2024г. №22-16-17/5456«О направлении методических рекомендаций» (вместе с «Методическими рекомендациями по разработке и реализации дополнительных общеразвивающих программ (включая разноуровневые и модульные), «Методическими рекомендациями по разработке и экспертизе качества авторских дополнительных общеразвивающих программ»).

36. Устав ГБОУ «ДАТ «Солнечный город», его локальные акты.

Актуальность

Современная компьютерная индустрия активно развивается и оказывает значительное влияние на экономику, культуру социальную сферу человечества. Современные дети не только являются активными потребителями компьютерных программ, но и могут становиться их создателями. Так , живущий во Вьетнаме , Дамир Булдаков разработал расширение на языке JavaScript, которое позволило играм, созданным на образовательном языке Scratch, легко подключаться к платформе Playgama. Плагин оказался настолько качественным, что компания Playgama официально включила его в свой SDK (набор инструментов для разработчиков) и выплатила юному автору денежную премию. Этот пример наглядно демонстрирует, что при наличии качественного инструментария и правильно выстроенной мотивации дети способны создавать востребованные профессиональным сообществом программные продукты, начиная с самых ранних этапов обучения.

Именно на такой переход от позиции «потребителя» к позиции «создателя» нацелена данная образовательная программа. Используя обучающие среды, как прототип профессиональной разработки, затем переходя к более сложным языкам и инструментам, курс выстраивает трамплин от визуального программирования к индустриальному кодированию, что особенно актуально в контексте подготовки будущих инженеров.

Таким образом, программа носит пропедевтический характер, способствуя развитию инженерного мышления через игровую деятельность и разработку программ. Разработка игр позволяет формировать представления о программном обеспечении компьютера, его разработке и использовании программ в реальной жизни.

Курс разработан в соответствии с требованиями подготовки к участию в Национальной технологической олимпиаде Junior по дисциплине «Технологии и компьютерные игры». В течение первого года обучающиеся осваивают основы гейм-дизайна и базовое программное обеспечение, необходимое для реализации видеоигр. Во втором году программы предусмотрена возможность участия в треке Олимпиады с последующей доработкой этапов проектов в рамках кружковой деятельности.

Новизна программы

Программа сочетает практико-ориентированный подход с элементами визуализации, что делает обучение интерактивным и доступным даже для начинающих.

Особое внимание уделяется:

- алгоритмизации (пошаговое выполнение команд);
- творческому программированию (рисование фигур, создание анимаций, мини-игр);
- развитию логики и аналитического мышления через решение задач.

Отличительные особенности программы

Игровой подход – использование графики Turtle делает обучение увлекательным.
Проектная деятельность – учащиеся создают собственные программы, что развивает креативность.
Связь с реальными технологиями – используемые интрументы востребованы в IT-сфере, что повышает мотивацию.
Гибкость обучения – программа адаптируется под уровень группы.

Педагогическая целесообразность заключается в том, что в ходе реализации программы используется деятельностный подход к воспитанию и развитию ребенка средствами

Программа способствует:

- **формированию основ** вычислительного мышления**;**
- **развитию** математических и логических навыков**;**
- **повышению интереса к** техническим дисциплинам**;**
- **подготовке к более сложным языкам программирования.**

Обучение строится на принципах наглядности, доступности и постепенного усложнения материала, что позволяет каждому ребенку осваивать программу в комфортном темпе.

Адресат программы: обучающиеся 11–15 лет без предварительной подготовки.

Срок реализации и объем программы: 2 года, 288 часов.

Режим занятий: 2 раза в неделю по 2 академических часа (40 мин.).

Наполняемость группы: 10 – 12 человек.

Форма обучения: очная

Форма занятий: лекция, тренинг, тестирование, презентация, ролевые игры, защита проекта.

ЦЕЛЬ ПРОГРАММЫ

Цели программы: ранняя профессиональная ориентация обучающихся, повышение престижа инженерных и научных профессий, подготовка кадрового резерва для глобального технологического лидерства России в области информационных технологий.

Задачи программы:

Личностные задачи:

- научить обучающихся использовать компьютер в повседневной работе (искать и анализировать информацию в Сети Интернет);
- научить обучающихся выполнять проектные и исследовательские работы;
- развить личностные качества в процессе коллективной деятельности над проектом;
- способствовать социализации и адаптации обучающихся детей к жизни в обществе;
- обеспечить духовно-нравственное, гражданско-патриотическое, военно-патриотическое и трудовое воспитание детей и подростков.

Предметные задачи:

обучающийся должен научиться:

- работать на компьютере в различных текстовых редакторах и IDE;
- устанавливать и настраивать браузер в режиме разработчика;
- работать в сети Интернет (поиск, анализ, сохранение информации);
- разрабатывать и отлаживать программы на языке Python
- выполнять проектные исследовательские работы;

Метапредметные задачи:

Регулятивные:

научить обучающихся:

- ставить цель и задачи для реализации проектов;
- преобразовывать практическую задачу в познавательную самостоятельно или с помощью педагога;
- анализировать задачи, разбивая их на известные подзадачи и вычленяя новое;
- синтезировать из известного материала новую информацию, оценивая ее актуальность и востребованность;
- планировать собственную деятельность в соответствии с поставленной задачей и условиями ее реализации, действовать в соответствии с планом;
- контролировать и оценивать свои действия и вносить коррективы в их выполнение;
- пользоваться компьютерными источниками информации;
- организовывать свое рабочее (учебное) место;
- обучить навыкам соблюдения правил безопасности в процессе деятельности.

Познавательные:

научить обучающихся:

- подбирать и анализировать специальную литературу;
- осуществлять учебно-исследовательскую работу;
- понимать информацию, представленную в виде текста, рисунков, схем;
- оценивать актуальность и достоверность информации представленной в Сети Интернет.

СОДЕРЖАНИЕ ПРОГРАММЫ

Учебный план дополнительной общеразвивающей программы «IT- КВАНТУМ. Введение в компьютерную разработку»

1 год обучения

№ п/п	Наименование раздела, темы	Всего	Теория	Практика	Формы аттестации (контроля)
Модуль 1: Визуальные новеллы, 40 часов					
1.	Техника безопасности. Проектная деятельность в системе “Кванториумов”. Командообразование.	2	2	0	беседа, наблюдение
2.	Что такое игра. Жанры видеоигр	2	1	1	контроль результатов творческой деятельности
3.	Новеллы Концепция новеллы: идея, тема, логлайн	2	1	1	контроль результатов творческой деятельности
4.	Игровая команда. Разработка игры	2	1	1	контроль результатов творческой деятельности
5.	Игровой баланс	2	0	2	контроль результатов творческой деятельности
6.	Интерактивный сторителлинг	2	0	2	контроль результатов творческой деятельности
7.	Сеттинг и сюжет	2	0	2	контроль результатов творческой деятельности
8.	Структура повествования: линейные и нелинейные подходы	2	0	2	контроль результатов творческой деятельности
9.	Библия персонажей	2	1	1	контроль результатов творческой деятельности
10.	Конфликт в структуре сюжета	2	1	1	контроль результатов творческой деятельности
11.	Решения и последствия	2	0	2	контроль результатов творческой деятельности
12.	Типы игровых интерфейсов	2	0	2	контроль результатов творческой деятельности
13.	Прототипирование UI	2	0	2	контроль результатов творческой деятельности
14.	Реализация новеллы	2	0	2	контроль результатов творческой деятельности
15.	Возможности конструктора	2	0	2	контроль результатов творческой деятельности
16.	Поиск графических материалов из открытых источников	2	1	1	контроль результатов творческой деятельности
17.	Создание графических элементов (спрайтов и фонов)	2	1	1	контроль результатов творческой деятельности
18.	Сборка прототипа	2	0	2	контроль результатов творческой деятельности
19.	Рефлексия	2	1	1	контроль результатов творческой деятельности
20.	Защита игры	2	0	2	контроль результатов

					творческой деятельности
Модуль 2: Веб-технологии и программирование для визуальных новелл – 36 часов					
21.	HTML, основные теги (что такое тег, атрибут, базовые: div, p, img, ссылки).	2	2	0	беседа, наблюдение
22.	Monogatari. Работа с редактором кода (VS Code).	2	1	1	контроль результатов творческой деятельности
23.	Структура проекта Monogatari — какие файлы за что отвечают.	2	1	1	контроль результатов творческой деятельности
24.	Визуальное оформление с CSS: Шрифты, свойства шрифтов.	2	0	2	контроль результатов творческой деятельности
25.	Визуальное оформление с CSS: Фон, цвета	2	0	2	контроль результатов творческой деятельности
26.	Визуальное оформление с CSS: Стилизация окна диалога и кнопок.	2	0	2	контроль результатов творческой деятельности
27.	Визуальное оформление с CSS: Основы анимаций (transition, transform).	2	1	1	контроль результатов творческой деятельности
28.	Изображения и звук: как добавлять фоны, персонажей, музыку в Monogatari	2	0	2	контроль результатов творческой деятельности
29.	JavaScript — основы: Данные, типы данных, переменные	2	0	2	контроль результатов творческой деятельности
30.	JavaScript — основы: Арифметика.	2	0	2	контроль результатов творческой деятельности
31.	JavaScript — основы: Логика. Условный оператор (if/else).	2	0	2	контроль результатов творческой деятельности
32.	JavaScript — основы: Повторение. Циклы (for, while).	2	0	2	контроль результатов творческой деятельности
33.	JavaScript — основы: Массивы. Объекты	2	1	1	контроль результатов творческой деятельности
34.	JavaScript — основы: Объекты HTML (DOM). События	2	1	1	контроль результатов творческой деятельности
35.	Реализация игровой логики средствами JavaScript	2	0	2	контроль результатов творческой деятельности
36.	Сборка прототипа	2	1	1	контроль результатов творческой деятельности
37.	Рефлексия (что получилось, что нет, как улучшить).	2	0	2	контроль результатов творческой деятельности
38.	Защита игры (презентация перед аудиторией).	2	2	0	Защита проекта

Модуль 3: Классическое программирование на Python. Основы алгоритмов – 36 часов					
39.	Язык программирования Место Python в семье языков.	2	2	0	контроль результатов творческой деятельности
40.	Python — основы: Данные, типы данных, переменные	2	1	1	контроль результатов творческой деятельности
41.	Python — основы: Арифметика.	2	1	1	контроль результатов творческой деятельности
42.	Python — основы: Логика. Условный оператор (if/else).	2	0	2	контроль результатов творческой деятельности
43.	Python — основы: Повторение. Циклы (for, while).	2	0	2	контроль результатов творческой деятельности
44.	Python — основы: Списки, кортежи, множества, словари.	2	0	2	контроль результатов творческой деятельности
45.	Python — основы: Стиль написания программ. Идеомы Python.	2	1	1	контроль результатов творческой деятельности
46.	Алгоритмы: Сложность алгоритма.	2	0	2	контроль результатов творческой деятельности
47.	Алгоритмы: Базовый поиск	2	0	2	контроль результатов творческой деятельности
48.	Алгоритмы: Простые сортировки	2	0	2	контроль результатов творческой деятельности
49.	Алгоритмы: Рекурсия и «разделяй и властвуй»	2	0	2	контроль результатов творческой деятельности
50.	Алгоритмы: алгоритмы на структуры данных	2	0	2	контроль результатов творческой деятельности
51.	Алгоритмы: Обход графов	2	1	1	контроль результатов творческой деятельности
52.	Алгоритмы: Динамическое программирование (база)	2	1	1	контроль результатов творческой деятельности
53.	Решение задач базового уровня сложности	2	0	2	контроль результатов творческой деятельности
54.	Решение задач повышенного уровня сложности	2	1	1	контроль результатов творческой деятельности
55.	Решение задач олимпиадного уровня сложности	2	0	2	контроль результатов творческой деятельности
56.	Рефлексия	2	2	0	
Модуль 4: Разработка проекта с использованием искусственного интеллекта – 32 часа					
57.	Что такое проект	2	2	0	контроль результатов творческой деятельности
58.	Идея проекта	2	1	1	контроль результатов творческой деятельности
59.	Аудит цели проекта	2	1	1	контроль результатов творческой деятельности
60.	Предварительное техническое	2	0	2	контроль результатов

	задание.				творческой деятельности
61.	Роли в проекте. Распределение ролей в тендеме человек - ИИ	2	0	2	контроль результатов творческой деятельности
62.	Итеративная разработка	2	0	2	контроль результатов творческой деятельности
63.	Внутренние тесты и разработка через тестирование.	2	1	1	контроль результатов творческой деятельности
64.	Прототип. Что является минимальным работоспособным продуктом.	2	0	2	контроль результатов творческой деятельности
65.	Презентация прототипа	2	0	2	контроль результатов творческой деятельности
66.	Добавление функционала в проект.	2	0	2	контроль результатов творческой деятельности
67.	Интеграционное тестирование.	2	0	2	контроль результатов творческой деятельности
68.	Презентация продуктового результата проекта	2	0	2	контроль результатов творческой деятельности
69.	Документирование проекта	2	1	1	контроль результатов творческой деятельности
70.	Рефлексия полученного опыта.	2	1	1	контроль результатов творческой деятельности
71.	Защита проекта (презентация перед аудиторией).	2	0	2	контроль результатов творческой деятельности
72.	Обобщение знаний полученных по итогам курса.	2	0	2	контроль результатов творческой деятельности
	Итого:	**144**	**37**	**107**	

Содержание учебного плана дополнительной общеразвивающей программы «IT-квантум. Введение в компьютерную разработку» 1 год обучения

Модуль 1: Визуальные новеллы

Тема 1: Техника безопасности. Проектная деятельность в системе "Кванториумов"

Теория: Техника безопасности при работе с компьютерами. Кванториумы. Цели и задачи Кванториумов. Национальная технологическая инициатива. Роль IT-квантума. Чем работа в кванториуме отличается от обучения в школе или занятий в кружке. Проектная деятельность. Отличительные особенности проектной работы. Потоки создания ценности: продуктовый и образовательный. Виды проектов. Кейсы. Команда и её роль в проекте.

Практика: Дискуссия на заданную тему. Выбор проекта для реализации. Формулировка продуктового результата проекта. Опрос – «каких образовательных результатов планируют достичь учащиеся в ходе проектной деятельности». Тренинг командообразования.

Форма контроля: беседа, наблюдение.

Тема 2: Что такое игра. Жанры видеоигр

Теория: Определение понятия «игра». Игра как система правил, вызовов и наград. Классификация видеоигр по жанрам: экшен, стратегия, RPG, симуляторы, головоломки, визуальные новеллы и др. Гибридные жанры. Особенности каждого жанра: механики, целевая аудитория, примеры.

Практика: Интерактивная викторина «Угадай жанр по скриншоту и описанию». Работа в малых группах: выбрать 3 разных жанра и привести по 2 примера игр, объяснить, почему они относятся к этим жанрам.
Форма контроля: устный опрос, выполнение группового задания.

Тема 3: Новеллы. Концепция новеллы: идея, тема, логлайн
Теория: Что такое визуальная новелла: отличия от книги, фильма и классической RPG. Ключевые элементы: текст, графика, выборы, ветвления. Понятия идеи (что я хочу рассказать), темы (о чём эта история глубже), логлайна (одно-два предложения, передающих суть). Примеры удачных логлайнов известных новелл.
Практика: Мозговой штурм — каждый придумывает 3 идеи для будущей новеллы. Формулирует к одной из них тему и логлайн. Презентация своего логлайна группе, обратная связь.
Форма контроля: контроль результатов творческой деятельности

Тема 4: Игровая команда. Разработка игры
Теория: Роли в игровой команде: продюсер, геймдизайнер, сценарист, художник, программист, тестировщик, звукорежиссёр. Распределение обязанностей. Особенности инди-разработки (соло или малая команда, совмещение ролей). Этапы разработки игры: пре-продакшн, продакшн, тестирование, релиз, пост-релизная поддержка.
Практика: Проектная сессия — каждая команда (3-4 человека) распределяет роли для своего проекта. Составление дорожной карты проекта на ближайшие 4 недели. Инструменты для планирования
Форма контроля контроль результатов творческой деятельности

Тема 5: Игровой баланс
Теория: Что такое баланс в игре. Почему важно, чтобы игра была честной, но интересной. Виды баланса: сложность, темп, ресурсы, экономика. Понятия «кривая сложности», «онбординг», «туториал». Баланс в визуальных новеллах: влияние выборов, доступность всех веток, избегание «тупиков» и «мёртвых концов».
Практика: Проектирование простой системы баланса для своего прототипа (шкала отношений героев, очки действий).
Форма контроля: контроль результатов творческой деятельности

Тема 6: Интерактивный сторителлинг
Теория: Определение интерактивного сторителлинга. Отличие от линейного повествования. Способы вовлечения игрока: выбор, последствия, иллюзия свободы, моральные дилеммы. Принцип «agency» (способность игрока влиять на мир). Понятие «водяной знак» сюжета — общая нить, несмотря на ветвления.
Практика: Разбор конкретных сцен из новелл, где выбор игрока действительно меняет ход событий. Создание небольшой диалоговой сцены (5-7 реплик) с двумя вариантами ответа, каждый из которых ведёт к разной реакции персонажа.
Форма контроля: контроль результатов творческой деятельности

Тема 7: Сеттинг и сюжет
Теория: Сеттинг: время, место, законы мира (магия, технологии, социальный строй). Сюжет: последовательность событий. Как сеттинг влияет на сюжет и наоборот. Типы сеттингов: фэнтези, научная фантастика, постапокалипсис, современность, исторический и т.д. Конфликт как двигатель сюжета: внутренний и внешний.
Практика: Описание сеттинга для своего проекта (2-3 абзаца: где и когда, какие правила, кто живёт). Построение простой сюжетной арки из 5 точек (завязка, инцидент, развитие, кульминация, развязка). Взаимная проверка в парах.
Форма контроля: контроль результатов творческой деятельности

Тема 8: Структура повествования: линейные и нелинейные подходы
Теория: Линейное повествование (игрок не влияет на порядок событий). Нелинейное: ветвящийся сюжет, множественные концовки, открытый мир, хабовая система (общие узлы с разными выходами). Преимущества и недостатки каждого подхода. Сложность реализации нелинейных структур для автора и тестировщика. Понятие «дерево диалогов».
Практика: Построение схемы сюжета для своего прототипа: линейный, с одной развилкой или с несколькими концовками. Использование диаграммы в Miro или на бумаге. Анализ чужой схемы (работа в парах).
Форма контроля: контроль результатов творческой деятельности

Тема 9: Библия персонажей
Теория: Что такое «библия персонажа»: документ, описывающий внешность, характер, привычки, мотивацию, бэкстори, речевые особенности. Зачем нужна библия: для единообразия написания диалогов, для художников, для избегания противоречий. Примеры структур библии.
Практика: Создание библии для 2-3 ключевых персонажей своего проекта (имя, возраст, цель, слабости, фраза-характеристика, внешность). Презентация одного персонажа группе.
Форма контроля: контроль результатов творческой деятельности .

Тема 10: Конфликт в структуре сюжета
Теория: Природа конфликта: без конфликта нет истории. Типы конфликтов: человек против человека, человека против природы, человека против общества, человека против себя (внутренний). Конфликт как источник выборов в новелле. Эскалация конфликта. Разрешение конфликта.
Практика: определить главный конфликт своего проекта. Написать короткую сцену (5-10 реплик), где конфликт проявляется и требует от игрока решения (два варианта выбора). Обсуждение в группе: насколько выборы отражают суть конфликта.
Форма контроля: контроль результатов творческой деятельности.

Тема 11: Решения и последствия
Теория: Механика «причина-следствие» в визуальных новеллах. Скрытые и явные последствия. Краткосрочные (немедленная реакция) и долгосрочные (влияют на финал) последствия. Как проектировать последствия, чтобы они не были слишком предсказуемыми или, наоборот, случайными. Понятие «вес выбора».
Практика: К каждой из двух развилок своего сюжета прописать минимум одно краткосрочное и одно долгосрочное последствие. Заполнить таблицу «Выбор → Немедленный результат → Отсроченный результат».
Форма контроля: таблица последствий.

Тема 12: Типы игровых интерфейсов
Теория: UI (пользовательский интерфейс) в визуальных новеллах: окно диалога, кнопки выбора, меню сохранения, экран настроек, инвентарь (если есть), журнал сообщений. Типы интерфейсов: минималистичный, кастомный (стилизованный под сеттинг), с анимацией. Принципы usability: понятность, отзывчивость, отсутствие перегруза.
Практика: Анализ интерфейсов 2-3 известных новелл (назвать, что удачно, что нет). Создание эскиза UI для своего проекта на бумаге или в Figma: расположение текстового окна, кнопок, портретов персонажей.
Форма контроля: контроль результатов творческой деятельности .

Тема 13: Прототипирование UI
Теория: От эскиза к интерактивному прототипу. Инструменты: Figma, Adobe XD, даже PowerPoint. Создание кликабельного прототипа для проверки логики переходов. Важность прототипирования перед программированием.

Практика: В Figma создать кликабельный прототип главного экрана новеллы (окно диалога, 2 кнопки выбора, меню). Протестировать на соседе: понятно ли, куда нажимать? Внести правки.
Форма контроля: контроль результатов творческой деятельности

Тема 14: Реализация новеллы
Теория: Обзор способов создания визуальной новеллы: готовые движки (Ren'Py, Monogatari, TyranoBuilder), программирование на Unity/Godot, конструкторы (Visual Novel Maker). Критерии выбора: сложность, возможности, необходимость программирования. Структура типового проекта: папки со спрайтами, фонами, музыкой; файлы скриптов.
Практика: Установка выбранного движка (на примере Monogatari). Запуск демо-проекта. Обзор папок и файлов проекта. Создание первой тестовой сцены с двумя репликами.
Форма контроля: демонстрация запущенной сцены.

Тема 15: Возможности конструктора (на примере Monogatari / Ren'Py)
Теория: Базовые команды движка: показать текст, показать фон, показать спрайт, воспроизвести музыку, создать выбор. Хранение переменных (очки отношений). Работа с сохранениями. Плюсы и минусы конструктора.
Практика: В своём проекте реализовать сцену из 5-7 реплик с двумя выборами, каждый из которых меняет значение переменной (например, «доверие»). Проверить, что переменная запоминается и может быть выведена в текст.
Форма контроля: работающий фрагмент новеллы с ветвлением.

Тема 16: Поиск графических материалов из открытых источников
Теория: Лицензии: Public Domain, Creative Commons (CC0, CC BY и др.), бесплатные для некоммерческого использования. Сайты: OpenGameArt, Pixabay, Unsplash (для фонов), Itch.io (бесплатные ассеты). Правила указания авторства. Ограничения: нельзя брать арты из Pinterest без подтверждённой лицензии.
Практика: Каждый участник находит 3 фона и 2 спрайта персонажа под условно-бесплатной лицензией. Фиксирует ссылки и лицензии в таблице. Групповой просмотр, отбор материалов для своего проекта.
Форма контроля: таблица с источниками и лицензиями.

Тема 17: Создание графических элементов (спрайтов и фонов)
Теория: Основы пиксельной графики и векторной графики. Размеры спрайтов и фонов для новелл (обычно 1920x1080 для фона, спрайт персонажа – 800-1200 пикселей в высоту). Инструменты: GIMP, Krita, Aseprite (пиксель), Inkscape (вектор). Техники создания простого спрайта: силуэт, цвета, тени. Эмоции персонажа (смена выражения лица).
Практика: В GIMP или Krita нарисовать простого персонажа (например, рыцаря-манчкина) в одной позе и двумя эмоциями (нейтрально, радостно, зло). Или создать фон комнаты/подземелья. Для начинающих – доработать найденные свободные ассеты (изменить цвет, добавить детали).
Форма контроля: готовые графические файлы в папке проекта.

Тема 18: Сборка прототипа
Теория: Объединение сценария, графики, звука в единый работающий продукт. Что такое «прототип»: минимальная версия, демонстрирующая ключевую механику (ветвления, выборы, влияние решений). Итеративная сборка: сначала текст и ветки, потом добавление артов и музыки. Тестирование на себе и других.
Практика: Полная сборка прототипа визуальной новеллы на 5-10 минут геймплея (включает: титульный экран, 2-3 сцены, минимум 3 выбора, 2 концовки или значимые изменения в зависимости от выбора). Работа в команде: сценарист, художник, программист.
Форма контроля: исполняемый файл или ссылка на играбельную версию (HTML/Electron).

Тема 19: Рефлексия
Теория: Понятие рефлексии в проектном обучении. Зачем анализировать свою работу. Типы рефлексии: эмоциональная, деятельностная, содержательная. Вопросы для самоанализа: что получилось хорошо, что было самым трудным, что бы я сделал иначе, чему я научился.
Практика: Заполнение индивидуального листа рефлексии. Групповое обсуждение: каждый участник делится одним успехом и одной ошибкой. Коллективное составление списка рекомендаций для будущих проектов.
Форма контроля: заполненный лист рефлексии, выступление в группе.

Тема 20: Защита игры
Теория: Как презентовать игру: структура выступления (идея, механики, целевая аудитория, особенности реализации, демонстрация геймплея). Подготовка слайдов (3-5 слайдов) и тизер-видео/скринкаста. Критерии оценки: оригинальность, техническое исполнение, дизайн, соответствие теме, качество презентации.
Практика: Публичная защита прототипа перед наставниками и другими группами. Демонстрация игры (5-7 минут). Ответы на вопросы. Просмотр работ других команд, голосование в номинациях («Лучший сюжет», «Лучший визуальный стиль», «Самый манчкинский подход»).
Форма контроля: оценка по критериям, взаимооценка команд.

Модуль 2: Веб-технологии и программирование для визуальных новелл
Формат: пары занятий (2 урока по 40-45 минут), обязательная самостоятельная работа.

Тема 21: HTML, основные теги (понятие тега, атрибута, базовые элементы: div, p, img, ссылки)
Теория: Роль HTML в веб-технологиях. Структура HTML-документа: <!DOCTYPE html>, <html>, <head>, <body>. Понятия тега и атрибута. Блочные и строчные элементы. Описание базовых тегов: (контейнер), <p> (абзац), <img> (изображение, атрибуты src, alt), <a> (гиперссылка, атрибут href). Понятие семантической вёрстки.
Практика: Создание в VS Code простой HTML-страницы с заголовком, абзацем, изображением и ссылкой. Открытие в браузере. Экспериментирование с атрибутами.
Форма контроля: Проверка созданной страницы, устные вопросы по теории.
Тема 22: Monogatari. Работа с редактором кода (VS Code)
Теория: Обзор Monogatari как движка для визуальных новелл, основанного на HTML/CSS/JS. Установка и запуск проекта (через скачивание архива или npm). Знакомство с интерфейсом VS Code: обзор папок, открытие проекта, основные горячие клавиши, встроенный терминал. Рекомендуемые плагины для работы с Monogatari (Live Server, Prettier).
Практика: Скачивание и распаковка стартового шаблона Monogatari. Открытие папки проекта в VS Code. Запуск демо-игры с помощью Live Server или локального сервера.
Форма контроля: Демонстрация работающей демо-игры, настройка редактора.

Тема 23: Структура проекта Monogatari — назначение файлов
Теория: Обзор корневых папок и файлов: index.html (точка входа), js/script.js (сценарий игры), js/main.js (конфигурация и глобальные настройки), style/main.css (стили), assets/ (изображения, музыка, шрифты). Назначение файла assets/data/assets.json. Краткое знакомство с папкой engine/ (ядро движка, не подлежит изменению).
Практика: Открытие каждого из перечисленных файлов, поиск в них знакомых HTML-тегов, CSS-правил, JavaScript-конструкций. Изменение текста в script.js и перезагрузка игры.
Форма контроля: Составление схемы-памятки «Структура проекта Monogatari» (устно или графически).

Тема 24: Визуальное оформление с CSS. Шрифты, свойства шрифтов

Теория: Каскадные таблицы стилей (CSS): назначение и способы подключения к HTML. В Monogatari CSS уже подключён. Селекторы (тег, класс, идентификатор). Свойства шрифта: font-family, font-size, font-weight, font-style, color. Подключение внешних шрифтов (Google Fonts) и локальных шрифтов через @font-face. Единицы измерения в CSS: px, em, rem.
Практика: В файле style/main.css изменение шрифта текста диалогового окна (например, на Roboto или «Press Start 2P»). Изменение размера и цвета текста.
Форма контроля: Проверка внесённых изменений в CSS, объяснение выбора шрифта.

Тема 25: Визуальное оформление с CSS. Фон, цвета
Теория: Способы задания цвета в CSS: ключевые слова (red), шестнадцатеричный код (#ffcc00), rgb() / rgba(), hsl(). Свойство background-color. Фоновые изображения: background-image, background-size, background-position, background-repeat. Назначение фона для игрового окна в целом и для отдельных элементов интерфейса.
Практика: Замена фона главного меню и фоновой заливки диалогового окна. Добавление текстуры или градиента. Создание полупрозрачного фона для окна диалога с использованием rgba.
Форма контроля: Демонстрация фоновых стилей, проверка их отображения в игре.

Тема 26: Визуальное оформление с CSS. Стилизация окна диалога и кнопок
Теория: Определение классов Monogatari, отвечающих за интерфейс (например, .dialog, .text, .choice, .button). Стилизация рамок, теней (box-shadow), скругления углов (border-radius), внутренних и внешних отступов (padding, margin). Псевдоклассы для кнопок: :hover, :active. Оформление кнопок выбора.
Практика: Оформление окна диалога в стилистике выбранного сеттинга (например, имитация свитка или панели). Стилизация кнопок выбора (цвет, шрифт, визуальные эффекты при наведении).
Форма контроля: Скриншот или видеозапись стилизованного интерфейса.

Тема 27: Визуальное оформление с CSS. Основы анимаций (transition, transform)
Теория: Плавные переходы (transition) для изменения свойств элементов (цвета, размера, прозрачности). Трансформации (transform: scale, rotate, translate). Примеры анимации кнопок и появления текста. Анимация с помощью @keyframes. Принципы умеренного использования анимаций в визуальных новеллах.
Практика: Стилизация кнопок при наведении курсора (например, изменение масштаба, цвета фона). Анимация появления текста (плавное затухание и сдвиг). Создание пульсирующей иконки перехода.
Форма контроля: Проверка работающих анимаций в проекте.

Тема 28: Изображения и звук: добавление фонов, персонажей, музыки в Monogatari
Теория: Форматы изображений: PNG (поддержка прозрачности), JPG (для фонов). Размещение файлов в папках assets/scenes/ (фоны) и assets/characters/ (спрайты). Команды show scene и show character в script.js. Настройка позиционирования спрайтов. Рекомендуемые размеры: фон 1920×1080, спрайт — не более 1500 пикселей по высоте. Форматы звука: MP3, OGG. Папки assets/sounds/ и assets/music/. Команды play music, play sound, stop music. Рекомендации по оптимизации (сжатие изображений через TinyPNG, использование OGG для музыки).
Практика: Добавление в сцену фона и персонажа из ранее подобранных материалов. Добавление короткой фоновой музыки из бесплатных источников. Проверка корректной загрузки медиа.
Форма контроля: Сцена с фоном, персонажем и музыкальным сопровождением.

Тема 29: JavaScript — основы. Данные, типы данных, переменные
Теория: Роль JavaScript в Monogatari. Объявление переменных с помощью let и const. Примитивные типы данных: number, string, boolean, null, undefined. Оператор typeof.

Преобразование типов. Правила именования переменных (camelCase). Способы хранения данных в Monogatari: monogatari.storage().имяПеременной.
Практика: В консоли браузера (F12) объявление нескольких переменных, эксперименты с типами. В файле script.js объявление переменной playerName с присвоением значения и вывод в диалог через monogatari.storage().playerName.
Форма контроля: Устное объяснение разницы между let и const, демонстрация кода с переменными.

Тема 30: JavaScript — основы. Арифметические операции
Теория: Арифметические операторы: +, -, *, /, % (остаток от деления), ** (возведение в степень). Приоритет операций. Инкремент (++) и декремент (--). Составное присваивание (+=, -= и др.). Особенности работы с числами с плавающей точкой.
Практика: Написание функции или фрагмента кода, вычисляющего суммарный уровень персонажа (сила + ловкость + интеллект) с сохранением результата в переменную. Вывод результата в диалоговое окно.
Форма контроля: Проверка корректности вычислений, использование арифметики в игровом контексте.

Тема 31: JavaScript — основы. Логика. Условный оператор (if/else)
Теория: Логический тип данных (true / false). Операторы сравнения: ==, ===, !=, !==, >, <, >=, <=. Логические операторы && (И), || (ИЛИ), ! (НЕ). Конструкции if, else if, else. Тернарный оператор (? :).
Практика: В сценарий новеллы добавление проверки: если значение переменной (например, уровень силы) больше 10, то открывается одна ветка диалога, иначе — другая. Использование monogatari.storage().strength.
Форма контроля: Демонстрация работы условного ветвления в игре.

Тема 32: JavaScript — основы. Повторение. Циклы (for, while)
Теория: Назначение циклов. Синтаксис цикла for (итерация по счётчику). Цикл while (выполнение пока условие истинно). Операторы break и continue. Возможные ошибки (бесконечный цикл). Примеры применения: заполнение массива, перебор элементов.
Практика: Написание цикла, который пять раз выводит в консоль различные реплики. В контексте игры — создание цикла для начисления бонусов за несколько последовательных побед над монстрами. Дополнительное задание: вывод списка инвентаря с помощью цикла.
Форма контроля: Проверка понимания через задание

Тема 33: JavaScript — основы. Массивы. Объекты
Теория: Массивы: способы создания, доступ к элементам по индексу, свойство length. Базовые методы: push, pop, shift, unshift, splice, slice. Перебор массива с помощью цикла for. Объекты: структура «ключ-значение». Создание объекта через {}, доступ к свойствам через точку или квадратные скобки. Вложенные объекты.
Практика: Создание массива с именами монстров. Реализация случайного выбора монстра из массива при входе в комнату. Создание объекта player с полями hp, attack, inventory (массив предметов).
Форма контроля: Код, демонстрирующий работу с массивом и объектом в игровом контексте.

Тема 34: JavaScript — основы. Объекты HTML (DOM). События
Теория: DOM (Document Object Model) — представление HTML-документа в браузере. Поиск элементов с помощью document.getElementById, querySelector. Изменение содержимого и атрибутов элементов. События (click, mouseover, input) и обработчики событий через addEventListener. В Monogatari прямое манипулирование DOM применяется редко, но может быть полезно для создания кастомных меню.
Практика: Создание отдельного HTML-файла с кнопкой и текстовым полем; при нажатии на

кнопку текст изменяется. В проекте Monogatari — добавление кастомной кнопки «Показать статистику», которая выводит текущие значения переменных во всплывающее окно.
Форма контроля: Работающий интерактивный элемент вне основного движка или интегрированный в Monogatari.

Тема 35: Реализация игровой логики средствами JavaScript
Теория: Объединение изученных конструкций: переменные, условия, циклы, массивы, объекты. Проектирование игровых систем для «Манчкин-подобной» новеллы: генерация случайного числа (бросок кубика), расчёт бонусов от предметов, проверка уровней, механика «удар в спину». Использование `monogatari.storage()` для долговременного хранения данных между сценами. Создание вспомогательных функций (например, `rollDice()`).
Практика: Реализация мини-игры «Бой с монстром». Игрок нажимает кнопку «Атаковать», вычисляется случайный урон, уменьшается здоровье монстра. Результат каждого хода выводится в диалоговое окно. При победе выполняется переход к следующей сцене.
Форма контроля: Полностью работающий фрагмент игровой логики (бой, проверка навыков, выбор предмета).

Тема 36: Сборка прототипа
Теория: Требования к прототипу для демонстрации: продолжительность геймплея 5–10 минут, наличие 3–5 значимых выборов, 2 и более концовок или альтернативных путей развития сюжета. Этапы интеграции сценария, графики, звука и игровой логики. Тестирование на наличие ошибок (логические петли, неработающие переходы, неинициализированные переменные). Инструменты отладки: консоль браузера, `console.log`. Экспорт игры для демонстрации (локальный сервер, сборка в исполняемый файл через Electron).
Практика: Доработка командами своих прототипов по предоставленному чек-листу. Проведение тестирования силами соседней команды. Исправление критических ошибок. Подготовка демонстрационной версии.
Форма контроля: Готовый к показу прототип.

Тема 37: Рефлексия (анализ результатов и процесса работы)
Теория: Понятие рефлексии в проектном обучении. Анализ сильных и слабых сторон реализованного проекта, а также командного взаимодействия. Выявление наиболее сложных с точки зрения программирования этапов. Определение возможных улучшений архитектуры игры.
Практика: Заполнение рефлексивной карты по трём колонкам: «достижения», «проблемы», «идеи для улучшения». Групповое обсуждение: каждая команда делится одним техническим успехом и одной ошибкой. Коллективная выработка рекомендаций (типичные ошибки и способы их избегания).
Форма контроля: Письменная рефлексия, устное выступление от команды.

Тема 38: Защита игры (презентация перед аудиторией)
Теория: Структура презентации проекта: название, жанр, основная идея, целевая аудитория, описание ключевых особенностей реализации (использованные возможности HTML/CSS/JS). Демонстрация геймплея продолжительностью 3–5 минут с показом важнейших механик. Критерии оценки: техническая сложность, стабильность работы, дизайн интерфейса, оригинальность, качество устной презентации. Рекомендации по публичному выступлению.
Практика: Публичная защита прототипа перед наставниками и другими учебными группами. Каждая команда демонстрирует игру (5–7 минут) и отвечает на вопросы. Проведение зрительского голосования в номинациях (например, «Лучший сюжет», «Лучший код», «Самый манчкинский дизайн»).
Форма контроля: Итоговая оценка проекта, взаимооценка команд, заполнение наставником листа наблюдения.

Модуль 3: Классическое программирование на Python. Основы алгоритмов
Формат: пары занятий (2 урока по 40–45 минут), обязательная самостоятельная работа.

Тема 39: Язык программирования Python. Место Python в семье языков
Теория: История создания Python, философия языка (PEP 20 – The Zen of Python). Классификация языков программирования: компилируемые и интерпретируемые, высокоуровневые и низкоуровневые. Место Python среди других языков (C++, Java, JavaScript). Области применения Python: веб-разработка, анализ данных, машинное обучение, автоматизация, научные вычисления, образование. Преимущества и недостатки Python.
Практика: Установка интерпретатора Python (последней стабильной версии) и среды разработки (VS Code или PyCharm Community Edition). Написание первой программы: вывод строки «Hello, World!» в консоль. Запуск скрипта из командной строки и из IDE.
Форма контроля: Устный опрос: основные характеристики Python; демонстрация работающей программы.

Тема 40: Python — основы. Данные, типы данных, переменные
Теория: Понятие переменной в программировании. Правила именования переменных (PEP 8). Присваивание значения. Базовые типы данных: целые числа (int), числа с плавающей точкой (float), строки (str), булевый тип (bool). Определение типа переменной с помощью type(). Динамическая типизация в Python. Преобразование типов: int(), float(), str(). Консольный ввод и вывод: print() и input().
Практика: Написание скрипта, который запрашивает у пользователя имя и возраст, затем выводит приветствие и тип введённых данных. Эксперименты с преобразованием типов.
Форма контроля: Проверка письменного задания (3–5 небольших фрагментов кода на объявление переменных и преобразование типов).

Тема 41: Python — основы. Арифметика
Теория: Арифметические операторы: сложение (+), вычитание (-), умножение (*), деление (/), целочисленное деление (//), остаток от деления (%), возведение в степень (**). Приоритет операций, использование круглых скобок. Особенности арифметики с числами с плавающей точкой (погрешности вычислений). Модуль math и его основные функции (math.sqrt, math.floor, math.ceil).
Практика: Написание программы для вычисления площади круга, дискриминанта квадратного уравнения, среднего арифметического нескольких чисел. Использование math для более сложных вычислений.
Форма контроля: Проверка решения 3–4 вычислительных задач.

Тема 42: Python — основы. Логика. Условный оператор (if/else)
Теория: Логический тип bool, значения True и False. Операторы сравнения: ==, !=, >, <, >=, <=. Логические операторы: and, or, not. Конструкции ветвления: if, elif, else. Вложенные условные операторы. Тернарный условный оператор (выражение X if условие else Y).
Практика: Написание программы, определяющей, является ли год високосным. Программа для сравнения трёх чисел и нахождения наибольшего. Реализация простого калькулятора с выбором действия через if/elif/else.
Форма контроля: Демонстрация работоспособности кода с различными входными данными.

Тема 43: Python — основы. Повторение. Циклы (for, while)
Теория: Цикл while: синтаксис, условие продолжения, опасность бесконечного цикла. Операторы break и continue. Цикл for: итерация по последовательностям (строки, списки, диапазоны). Функция range(): различные варианты вызова (range(stop), range(start, stop), range(start, stop, step)). Вложенные циклы.
Практика: Написание программы для вычисления факториала числа (через for и через while).

Программа для вывода таблицы умножения. Алгоритм проверки, является ли число простым.
Форма контроля: Проверка реализаций циклических алгоритмов (3 задачи).

Тема 44: Python — основы. Списки, кортежи, множества, словари
Теория:
Списки (list**)**: создание, доступ по индексу, срезы ([start:stop:step]), основные методы (append, extend, insert, remove, pop, index, count, sort, reverse). Вложенные списки.
Кортежи (tuple**)**: неизменяемость, создание, доступ по индексу, методы (count, index).
Множества (set**)**: неупорядоченность, уникальность элементов, операции над множествами (объединение, пересечение, разность, симметрическая разность), методы (add, remove, discard, clear).
Словари (dict**)**: пары «ключ–значение», создание, доступ по ключу, методы (keys, values, items, get, pop, update).
Практика: Реализация программы для подсчёта частоты слов в предложении (с использованием словаря). Работа со списком студентов и их оценками (средний балл). Использование множества для удаления дубликатов из списка.
Форма контроля: Задание на использование каждой из структур данных.

Тема 45: Python — основы. Стиль написания программ. Идиомы Python
Теория: Рекомендации PEP 8: отступы, максимальная длина строки, пробелы вокруг операторов, именование переменных и функций (snake_case), констант (UPPER_CASE). Идиомы Python (Pythonic code): распаковка последовательностей (a, b = b, a), генераторы списков и словарей, использование enumerate() и zip(), конструкция with для работы с файлами. Документирование кода с помощью docstring.
Практика: Рефакторинг «некрасивого» кода в соответствии с PEP 8 и идиомами. Написание функции с docstring. Применение генератора списков для создания квадратов чисел. Использование enumerate в цикле для вывода элементов списка с индексами.
Форма контроля: Проверка стиля кода в написанных учениками программах, устный разбор идиом.

Тема 46: Алгоритмы. Сложность алгоритма
Теория: Понятие алгоритма, его свойства (конечность, детерминированность, массовость). Необходимость анализа эффективности. Временная и пространственная сложность. «О-большое» (Big O notation): определение, основные классы сложности: O(1) – константная, O(log n) – логарифмическая, O(n) – линейная, O(n log n) – линейно-логарифмическая, $O(n^2)$ – квадратичная, $O(2^n)$ – экспоненциальная. Примеры алгоритмов для каждого класса.
Практика: Анализ простых фрагментов кода (циклы, вложенные циклы) с определением их сложности. Сравнение времени выполнения двух алгоритмов на разных объёмах входных данных (экспериментально с помощью модуля time).
Форма контроля: Письменный разбор 5–6 фрагментов кода с указанием сложности.

Тема 47: Алгоритмы. Базовый поиск
Теория:
Линейный поиск: принцип работы, сложность O(n). Реализация на Python. Поиск минимального/максимального элемента.
Бинарный поиск: условие применимости (отсортированный массив), алгоритм «деления пополам», сложность O(log n). Итеративная и рекурсивная реализации.
Практика: Написание функции линейного поиска для нахождения индекса заданного элемента. Реализация бинарного поиска на отсортированном списке. Сравнение времени работы обоих алгоритмов на больших массивах.
Форма контроля: Проверка реализованных функций поиска.

Тема 48: Алгоритмы. Простые сортировки

Теория: Задача сортировки. Классификация алгоритмов сортировки: устойчивые и неустойчивые, «на месте» и требующие дополнительной памяти.
Сортировка пузырьком (Bubble sort): описание, сложность $O(n^2)$, оптимизация с флагом обмена.
Сортировка вставками (Insertion sort): описание, эффективна на почти отсортированных данных, сложность $O(n^2)$ в худшем случае.
Сортировка выбором (Selection sort): поиск минимума и перемещение в начало.
Практика: Реализация каждого из трёх алгоритмов сортировки. Тестирование на случайных списках разного размера, измерение времени выполнения. Визуализация процесса сортировки (вывод промежуточных состояний).
Форма контроля: Демонстрация работающих функций сортировки, анализ их эффективности.

Тема 49: Алгоритмы. Рекурсия и «разделяй и властвуй»
Теория: Понятие рекурсии: базовый случай и рекурсивный шаг. Рекурсивные функции на Python. Риски: глубина рекурсии, переполнение стека (sys.setrecursionlimit). Классические примеры: вычисление факториала, чисел Фибоначчи (наивная рекурсия и её недостатки). Парадигма «разделяй и властвуй» (Divide and Conquer). Алгоритм **сортировки слиянием (Merge sort)**: описание, разделение на подмассивы, слияние, сложность O(n log n).
Практика: Написание рекурсивной функции для вычисления факториала и чисел Фибоначчи. Реализация сортировки слиянием. Сравнение времени работы сортировки слиянием с простыми сортировками на больших данных.
Форма контроля: Проверка рекурсивных функций и корректности сортировки слиянием.

Тема 50: Алгоритмы. Алгоритмы на структурах данных
Теория:
Стек (Stack): LIFO, реализация на Python с использованием списка (append, pop). Примеры применения: проверка сбалансированности скобок, обратная польская запись.
Очередь (Queue): FIFO, использование collections.deque для эффективной работы с обоими концами.
Связный список: односвязный и двусвязный (теоретическое понятие).
Хеш-таблица: принцип работы, разрешение коллизий. В Python это словарь (dict).
Практика: Реализация проверки баланса скобок с помощью стека. Реализация очереди для задачи обработки заявок. Использование deque для организации очереди.
Форма контроля: Демонстрация работы со стеком и очередью, объяснение выбора структуры данных для конкретной задачи.

Тема 51: Алгоритмы. Обход графов
Теория: Основные понятия теории графов: вершины, рёбра, ориентированные и неориентированные графы, взвешенные графы. Способы представления графов в программе: матрица смежности, список смежности (словарь списков).
Поиск в глубину (DFS): рекурсивная и итеративная реализации (стек).
Поиск в ширину (BFS): использование очереди. Нахождение кратчайшего пути в невзвешенном графе.
Практика: Реализация графа с использованием списка смежности. Написание DFS для обхода всех вершин. Реализация BFS для нахождения кратчайшего расстояния между двумя вершинами.
Форма контроля: Проверка реализованных обходов, демонстрация на небольшом графе.

Тема 52: Алгоритмы. Динамическое программирование (база)
Теория: Идея динамического программирования: разбиение задачи на подзадачи, запоминание результатов для повторного использования (мемоизация). Сравнение с наивной рекурсией на примере чисел Фибоначчи. Принципы оптимальности для подзадач.
Мемоизация «сверху вниз»: рекурсия с кэшированием (functools.lru_cache).

Табуляция «снизу вверх»: заполнение массива (DP-таблицы).
Практика: Реализация чисел Фибоначчи с мемоизацией и табуляцией. Решение классической задачи «Кубики» (найти количество способов добраться до вершины лестницы) или задачи о наборе монет (минимальное количество монет для сдачи).
Форма контроля: Представление и сравнение двух подходов ДП, анализ эффективности.

Тема 53: Решение задач базового уровня сложности
Теория: Обзор типов задач, которые считаются базовыми: работа с числами (простые числа, делители, палиндромы), строки (анаграммы, регистр, удаление символов), списки (сумма элементов, поиск дубликатов, срезы). Формулировка требований к решению: входные данные, выходные данные, ограничения.
Практика: Решение 4–5 задач из сборников (например, с сайта Codewars уровень 8 kyu или 7 kyu, или с acmp.ru номера 1–50). Задачи разбираются индивидуально или в парах. Проводится обсуждение различных подходов.
Форма контроля: Проверка оформленных решений (файлы .py). Устный разбор сложных моментов.

Тема 54: Решение задач повышенного уровня сложности
Теория: Признаки задач повышенной сложности: комбинирование нескольких структур данных, необходимость выбора эффективного алгоритма (не квадратичного), задачи на рекурсию и перебор, обработка больших входных данных.
Практика: Решение 3–4 задач из олимпиадных источников (например, Codeforces рейтинг 1000-1200, acmp.ru номера 100–300). Примеры: задача на поиск подстроки, задача на динамическое программирование начального уровня, задача на работу с графами (поиск компонент связности).
Форма контроля: Проверка решений, проведение мини-соревнования по времени (кто быстрее решит одну задачу).

Тема 55: Решение задач олимпиадного уровня сложности
Теория: Особенности олимпиадных задач: нестандартные условия, многомерные структуры данных, оптимизация по времени и памяти, применение изученных алгоритмов (бинарный поиск, ДП, BFS/DFS, рекурсия). Понятие «жадные алгоритмы» (краткое введение).
Практика: Разбор и решение 2–3 задач уровня всероссийской олимпиады (не самые сложные варианты). Примеры: задача на поиск кратчайшего пути во взвешенном графе (алгоритм Дейкстры — ознакомительно), задача на динамическое программирование на двумерном массиве («Черепашка»). Реализация в парах с последующей защитой решения.
Форма контроля: Экспертная оценка алгоритмической правильности и эффективности решений.

Тема 56: Рефлексия
Теория: Анализ прогресса в изучении Python и алгоритмов. Какие темы были наиболее сложными? Какие алгоритмы требуют дополнительного повторения? Оценка собственного уровня: от написания простых скриптов до решения олимпиадных задач. Планирование дальнейшего обучения: ресурсы (Codeforces, LeetCode, Stepik, «Типичный программист»), геймджемы, хакатоны.
Практика: Заполнение индивидуальной рефлексивной карты (достижения, пробелы, цели). Коллективное обсуждение: обмен опытом решения задач. Создание персональной дорожной карты «Куда двигаться после курса».
Форма контроля: Защита дорожной карты перед наставником; устная беседа.

Модуль 4: Разработка проекта с использованием искусственного интеллекта

Тема 57: Что такое проект

Теория: Определение понятия «проект» в контексте IT-разработки. Отличительные признаки проекта: целеполагание, ограниченность во времени, уникальность результата, ресурсные ограничения. Типы проектов по сложности, продолжительности и составу команды. Примеры успешных проектов в области создания игр и приложений.
Практика: Анализ 2–3 примеров проектов (из материалов Кванториума или открытых источников) на предмет наличия признаков проекта. Групповое обсуждение: чем проект отличается от «просто написания кода».
Форма контроля: Устный опрос, составление схемы признаков проекта.

Тема 58: Идея проекта
Теория: Источники идей: личный опыт, наблюдение, потребности рынка, анализ существующих решений. Методы генерации идей: мозговой штурм, майндмэппинг, метод «проблема – решение». Критерии отбора идей: новизна, реализуемость, полезность, соответствие техническим навыкам команды.
Практика: В командах (2–3 человека) проведение мозгового штурма для генерации не менее 5 идей проектов. Выбор одной идеи для дальнейшей разработки. Краткое устное обоснование выбора.
Форма контроля: Зафиксированный список идей, выбор основной идеи с обоснованием.

Тема 59: Аудит цели проекта
Теория: Понятие цели проекта. Техника SMART: цель должна быть конкретной, измеримой, достижимой, релевантной и ограниченной по времени. Отличие цели от задачи. Аудит цели: проверка на реалистичность, оценка ресурсов, выявление рисков.
Практика: Формулировка цели проекта по SMART. Проведение аудита цели внутри команды (каждый проверяет цель соседней команды). Корректировка цели по результатам аудита.
Форма контроля: Письменная формулировка цели проекта, прошедшая аудит.

Тема 60: Предварительное техническое задание (ТЗ)
Теория: Назначение технического задания. Структура предварительного ТЗ: назначение и цели, функциональные требования, нефункциональные требования (производительность, безопасность, платформа), ограничения и допущения. Пример ТЗ для небольшого программного продукта.
Практика: Составление предварительного технического задания на выбранный проект (объёмом 1–2 страницы). Включение требований к пользовательскому интерфейсу, к данным, к среде исполнения.
Форма контроля: Проверка наставником полноты и ясности ТЗ.

Тема 61: Роли в проекте. Распределение ролей в тандеме «человек – ИИ»
Теория: Классические роли в IT-проекте: менеджер, аналитик, разработчик, тестировщик, документатор. Особенности работы с инструментами искусственного интеллекта (ChatGPT, GitHub Copilot, Midjourney и др.) как с «виртуальными помощниками». Распределение ответственности: человек ставит задачу, контролирует результат, принимает итоговые решения; ИИ генерирует варианты, предлагает решения, выполняет рутинные операции.
Практика: Определение ролей внутри команды (какой участник за что отвечает). Составление схемы «Кто принимает решения по каким вопросам с участием ИИ». Назначение одного участника ответственным за работу с ИИ-инструментами.
Форма контроля: Схема распределения ролей и ответственности.

Тема 62: Итеративная разработка
Теория: Понятие итерации (цикла) в разработке. Сравнение каскадной (водопадной) и итеративной моделей. Преимущества итеративного подхода: быстрая обратная связь, снижение рисков, возможность изменять требования. Длина итерации (спринта) в учебном проекте (3–5 дней).

Практика: Планирование итераций для своего проекта: определение минимального набора функций для первой итерации, второй и т.д. Составление графика итераций с указанием дат завершения каждой.
Форма контроля: План итераций (таблица с датами и перечнем реализуемых функций).

Тема 63: Внутренние тесты и разработка через тестирование (TDD)
Теория: Понятие тестирования программного обеспечения. Внутренние тесты (силами разработчиков). Разработка через тестирование (Test-Driven Development, TDD): цикл «красный – зелёный – рефакторинг». Написание простых модульных тестов (unit tests) для Python (библиотека unittest или pytest).
Практика: Выбор одного модуля проекта (например, функция расчёта бонуса в игре). Написание теста, затем реализация функции, проходящей тест. Ознакомление с запуском тестов и анализом результатов.
Форма контроля: Демонстрация теста и функции, прошедшей тестирование.

Тема 64: Прототип. Минимальный работоспособный продукт (MVP)
Теория: Определение прототипа и минимального работоспособного продукта (Minimum Viable Product, MVP). Отличия: прототип может не иметь полной функциональности, но демонстрирует ключевую идею; MVP – это первая рабочая версия, которую можно показывать пользователям. Критерии MVP для учебного проекта.
Практика: Определение состава MVP для своего проекта (какие функции обязательно должны быть реализованы к концу первой итерации). Создание прототипа (бумажного или цифрового) интерфейса или ключевой механики.
Форма контроля: Список требований к MVP, презентация прототипа.

Тема 65: Презентация прототипа
Теория: Цели презентации прототипа: получение обратной связи, проверка жизнеспособности идеи, уточнение требований. Структура презентации: проблема, решение, демонстрация ключевого экрана или сценария, вопросы к аудитории. Рекомендации по проведению (хронометраж 5 минут на команду).
Практика: Публичная презентация прототипов перед наставником и другими командами. Сбор обратной связи (письменно или устно). Обсуждение полученных замечаний и предложений.
Форма контроля: Оценка наставником качества презентации и полноты прототипа.

Тема 66: Добавление функционала в проект
Теория: Принципы добавления новых функций после создания MVP. Приоритизация: метод MoSCoW (must have, should have, could have, won't have). Управление изменениями: ведение списка задач (backlog), оценка трудозатрат. Использование ИИ для генерации кода новых функций.
Практика: Формирование backlog на 2–3 следующих итерации. Реализация одной дополнительной функции с использованием ИИ-ассистента (например, ChatGPT запрашивается фрагмент кода или алгоритм). Интеграция функции в проект.
Форма контроля: Рабочий код новой функции, демонстрация её работы.

Тема 67: Интеграционное тестирование
Теория: Отличие интеграционного тестирования от модульного. Проверка взаимодействия между компонентами проекта. Типичные проблемы при интеграции: несовместимость интерфейсов, ошибки передачи данных, сбои в работе связанных функций. Подходы к интеграционному тестированию: «снизу вверх», «сверху вниз», «большой взрыв».
Практика: Проведение интеграционного тестирования созданного проекта (тестирование целых сценариев: запуск игры, выполнение цепочки действий, достижение конца). Фиксация обнаруженных ошибок и их исправление.

Форма контроля: Отчёт о проведённом тестировании (список найденных и исправленных дефектов).

Тема 68: Презентация продуктового результата проекта
Теория: Отличия презентации продуктового результата от презентации прототипа. Акцент на завершённости, стабильности, пользовательской ценности. Структура: введение, демонстрация готового продукта, ключевые технические решения, полученные результаты и метрики. Подготовка демонстрационной среды (установка на ноутбук, запись видео).
Практика: Репетиция презентации в команде. Публичная демонстрация готового продукта (игра, приложение, веб-сервис) перед аудиторией. Ответы на вопросы.
Форма контроля: Оценка наставником соответствия продукта утверждённому ТЗ и качеству демонстрации.

Тема 69: Документирование проекта
Теория: Виды документации: пользовательская (инструкция по установке и использованию), техническая (описание архитектуры, API, базы данных), проектная (ТЗ, план итераций, результаты тестирования). Требования к документации: полнота, понятность, актуальность. Использование ИИ для помощи в составлении документации.
Практика: Составление двух документов: «Руководство пользователя» (1 страница) и «Краткое техническое описание» (1–2 страницы). Включение скриншотов и примеров кода.
Форма контроля: Проверка наставником качества и полноты документации.

Тема 70: Рефлексия полученного опыта
Теория: Анализ процесса разработки с использованием ИИ. Какие задачи были эффективно решены с помощью ИИ, а какие потребовали вмешательства человека. Оценка продуктивности работы в тандеме. Выявление приобретённых навыков: формулирование запросов (промтов) для ИИ, критическая оценка результатов, интеграция сгенерированного кода.
Практика: Заполнение индивидуального опросника рефлексии по образцу (5 вопросов: что получилось, что нет, чему научился, что было неожиданным, как изменилось отношение к ИИ). Групповое обсуждение успехов и трудностей.
Форма контроля: Письменный отчёт по рефлексии.

Тема 71: Защита проекта (презентация перед аудиторией)
Теория: Финальная защита проекта – итоговое мероприятие курса. Структура выступления: проблема, решение, технические детали, демонстрация (видео или живая), вклад ИИ в разработку, планы развития. Критерии оценки: полнота реализации ТЗ, качество кода и документации, умение отвечать на вопросы, использование ИИ.
Практика: Публичная защита проекта перед комиссией (наставники, приглашённые эксперты, другие группы). Длительность защиты – 7–10 минут на команду, включая демонстрацию и вопросы.
Форма контроля: Итоговая оценка проекта по согласованным критериям.

Тема 72: Обобщение знаний, полученных по итогам курса
Теория: Связь всех трёх модулей (проектирование новеллы, веб-технологии, классическое программирование, разработка с ИИ). Формирование целостной картины разработки программного продукта. Вопросы для итогового собеседования: жизненный цикл проекта, роль документации, виды тестирования, использование ИИ.
Практика: Проведение итоговой тестовой работы (20 вопросов с выбором ответа) или устного собеседования. Подведение итогов курса, вручение сертификатов/дипломов.
Форма контроля: Результаты тестирования или собеседования.

По окончанию 1 года обучения учащиеся будут знать и уметь:

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ

Личностные результаты:

- Обучающийся умеет использовать компьютер и цифровые инструменты (включая ИИ) для решения прикладных проектных задач, поиска и критического анализа информации в сети Интернет.
- Обучающийся выполняет проектные и исследовательские работы в области разработки визуальных новелл, веб-приложений, алгоритмических задач и программных продуктов с использованием искусственного интеллекта.
- У обучающихся развиты личностные качества в процессе коллективной деятельности над проектом: ответственность, инициативность, дисциплинированность, способность доводить начатое до конца.
- Сформирована потребность в сотрудничестве со сверстниками, доброжелательное отношение к членам команды, бесконфликтное поведение, умение прислушиваться к мнению товарищей по команде и аргументированно отстаивать свою позицию.
- Осуществлено духовно-нравственное, гражданско-патриотическое, военно-патриотическое и трудовое воспитание через тематику проектных заданий (в том числе создание новелл на историческую или социально значимую тему) и осознание ценности интеллектуального труда.
- Сформировано уважительное отношение к интеллектуальной собственности, понимание лицензий (Creative Commons, Open Source) и правил использования готовых графических, музыкальных и программных материалов, а также результатов работы ИИ.

Предметные результаты:

Обучающийся научился:

- **Работать на компьютере** в различных текстовых редакторах (VS Code, PyCharm, блокнот) и интегрированных средах разработки, настраивать их под свои задачи.
- **Устанавливать и настраивать браузер** в режиме разработчика (инструменты разработчика Chrome/Firefox) для отладки HTML/CSS/JS, анализа сетевых запросов и проверки адаптивности интерфейса.
- **Работать в сети Интернет** эффективно: осуществлять поиск информации по ключевым словам, фильтровать результаты, проверять источники на достоверность, сохранять и систематизировать найденные материалы (изображения, звуки, библиотеки кода, документацию).
- **Разрабатывать и отлаживать веб-страницы и программы** на языках **HTML**, **CSS**, **JavaScript** (в том числе с использованием движка визуальных новелл Monogatari): создавать структуру страницы, стилизовать интерфейс, прописывать диалоги и выборы, использовать переменные, условия, события и анимации.
- **Создавать и обрабатывать растровые и векторные изображения** (спрайты персонажей, фоны) в графических редакторах (GIMP, Krita, Inkscape) и подготавливать их для интеграции в игру.
- **Писать программы на языке Python** (переменные, типы данных, условные операторы, циклы, списки, словари, функции) и применять их для решения вычислительных и алгоритмических задач.
- **Оценивать сложность алгоритмов** (O-нотация), реализовывать базовые алгоритмы поиска (линейный, бинарный), сортировки (пузырьком, вставками, слиянием), рекурсивные функции, обход графов (DFS, BFS), простейшие алгоритмы динамического программирования.
- **Формулировать техническое задание**, разбивать проект на итерации, использовать минимально работоспособный продукт (MVP) и прототипирование, документировать результаты (руководство пользователя, техническое описание).
- **Использовать инструменты искусственного интеллекта** (чат-боты, генераторы кода, графические нейросети) как вспомогательные средства при проектировании, написании кода,

генерации идей, составлении документации и тестировании, при этом критически оценивая результаты их работы.

• **Выполнять проектные и исследовательские работы** в команде: от генерации идеи и постановки цели до публичной защиты продукта, включая самоанализ (рефлексию) и планирование дальнейшего развития.

Метапредметные результаты:
Обучающийся умеет:

• **Ставить цель и задачи** для реализации проектов на основе анализа потребностей и имеющихся ресурсов, используя методы SMART.

• **Преобразовывать практическую задачу** (создание игры, приложения, алгоритма) в познавательную (как это работает, какие технологии выбрать) — самостоятельно или с помощью педагога.

• **Анализировать задачи** по разработке программного продукта, разбивая их на известные подзадачи (отрисовка интерфейса, написание скрипта, отладка) и выделяя новое, требующее дополнительного изучения.

• **Синтезировать из известного материала новую информацию** (комбинировать команды CSS, функции JavaScript, библиотеки Python), оценивая её актуальность и востребованность для своего проекта.

• **Планировать собственную деятельность** (составлять дорожную карту проекта, график итераций, список задач на спринт) в соответствии с поставленной задачей и условиями её реализации, действовать в соответствии с планом.

• **Контролировать и оценивать свои действия** (проводить модульное и интеграционное тестирование, анализировать ошибки, измерять время выполнения алгоритмов) и вносить коррективы в их выполнение.

• **Пользоваться компьютерными источниками информации** (официальная документация, форумы, базы знаний, онлайн-курсы, генеративные ИИ-модели) для решения возникающих технических проблем и расширения кругозора.

• **Организовывать своё рабочее (учебное) место** (удобное расположение окон редактора, браузера, консоли, системы контроля версий) и поддерживать порядок в файловой структуре проекта.

• **Соблюдать правила безопасности** при работе за компьютером, в сети Интернет (защита персональных данных, проверка ссылок, использование лицензионного ПО) и при использовании ИИ-сервисов.

Познавательные Обучающийся умеет:

• **Подбирать и анализировать специальную литературу** (документацию по HTML/CSS/JS, Python, Monogatari, алгоритмам), выделяя главное и применимое к своей задаче.

• **Осуществлять учебно-исследовательскую работу** (сравнение алгоритмов сортировки по времени работы, анализ эффективности разных подходов к хранению игровых данных, тестирование влияния размера изображений на производительность новеллы).

• **Понимать информацию, представленную в разных формах**: текст (техническое задание, баги-репорты), рисунки (схемы ветвлений сюжета, диаграммы классов, скринкасты), схемы (алгоритмические блок-схемы, архитектура проекта, дерево переходов), таблицы (Big O сложностей, результаты тестирования).

• **Оценивать актуальность и достоверность информации**, представленной в сети Интернет: отличать устаревшую документацию от современной, проверять репутацию источников, сверять данные с официальными сайтами и несколькими независимыми ресурсами, распознавать сгенерированный ИИ текст, требующий проверки.

• **Умеет работать в команде, распределяя роли** (сценарист, художник, программист, тестировщик), согласовывать интерфейсы между компонентами, проводить ретроспективы и давать обратную связь товарищам.

• **Способен публично представить результаты проекта** (защита прототипа, продуктового результата), аргументированно ответить на вопросы аудитории и комиссии.

СОДЕРЖАНИЕ ПРОГРАММЫ
Учебный план дополнительной общеразвивающей программы
«IT- КВАНТУМ. Введение в компьютерную разработку»
2 год обучения

№ п/п	Наименование раздела, темы	Всего	Теория	Практика	Формы аттестации (контроля)
Модуль 1. Командная разработка и управление IT-проектами – 36 часов					
1.	Введение в командную разработку. Жизненный цикл IT-проекта.	2	2	0	беседа, наблюдение
2.	Роли в команде: продюсер, PM, разработчик, тестировщик, дизайнер. Распределение ролей.	2	1	1	контроль результатов творческой деятельности
3.	Системы управления проектами: Обзор «Битрикс24». Доска в электронных таблицах. Создание доски задач.	2	1	1	контроль результатов творческой деятельности
4.	Командная работа: мозговой штурм, определение идеи сквозного проекта. Сервис Яндекс.Доски	2	1	1	контроль результатов творческой деятельности
5.	Agile и Scrum: спринты, бэклог, ежедневные встречи. Создание бэклога и планирование недельного спринта	2	1	1	контроль результатов творческой деятельности
6.	Git: установка, базовые команды (init, add, commit, status, log). ветки (branch, checkout, merge), разрешение конфликтов.	2	1	1	контроль результатов творческой деятельности
7.	Разработка спринта 1: распределение задач, работа в ветках Git. Ежедневные стендапы (имитация + дистант). Ревью кода внутри команды.	2	1	1	контроль результатов творческой деятельности
8.	Работа с удалённым репозиторием: Pull request и code review. Баг-трекинг: оформление issue, метки, приоритеты.	2	0	2	контроль результатов творческой деятельности

9.	Разработка спринта 2: добавление функционала, решение конфликтов. Тестирование и приёмка: оформление багов, приоритизация.	2	1	1	контроль результатов творческой деятельности
10.	CI/CD: знакомство. Автоматическая сборка и тесты.	2	0	2	наблюдение
11.	Разработка спринта 3: доработка по фидбеку. Интеграционное тестирование и финальная сборка.	2	0	2	контроль результатов творческой деятельности
12.	Техническая документация: README, Wiki, архитектурные схемы. Написание пользовательской документации и руководства	2	0	2	контроль результатов творческой деятельности
13.	Демонстрация прототипа. Сбор обратной связи. Ретроспектива.	2	0	2	защита этапа проекта
14.	Внесение изменений в проект на основании обратной связи. Рефакторинг кода. Код-ревью. Обсуждение Pull Request в GitVerse. Pinning tests	2	0	2	контроль результатов творческой деятельности
15.	AI-помощь в модификации кода: анализ текущей кодовой базы, комментирование, и выявление точек модификации согласно техническому заданию; генерация конкретных изменений (фрагменты кода, правки структур данных, оптимизация алгоритмов) с пояснением логики; проверка изменений на синтаксические и логические ошибки, предложение тестовых сценариев; рефакторинг и унификация стиля для поддержки сопровождаемости; документирование внесённых правок и формирование итогового отчёта.	2	0	2	контроль результатов творческой деятельности
16.	Презентация готового продукта (сквозной проект).	2	0	2	защита проекта
17.	Итоговая аттестация по модулю: защита портфолио (доска задач, репозиторий, документация).	2	0	2	защита проекта
18.	Рефлексия образовательных результатов	2	2	0	беседа
Модуль 2. Событийное программирование в среде «Берлога» + основы 3D-моделирования – 36 часов					

№ п/п	Наименование раздела, темы	Всего	Теория	Практика	Формы аттестации (контроля)
19.	Знакомство с игрой «Берлога: Защита пасеки». Разбор структуры готового игрового проекта.	2	1	1	беседа, наблюдение
20.	Изучение логики поведения объектов. Создание схемы игрового процесса с выделением ключевых элементов логики.	2	0	2	контроль результатов творческой деятельности
21.	Машины состояний. Понятие состояния. Создание системы с простыми состояниями.	2	1	1	контроль результатов творческой деятельности
22.	Машины состояний. Переходы. Создание схемы машины состояний для персонажа.	2	0	2	контроль результатов творческой деятельности
23.	Машины состояний. События. Обработка событий в игровой логике. Роль триггеров и условий.	2	1	1	контроль результатов творческой деятельности
24.	Реализация логики смены состояний на практике (в среде «Берлога»).	2	0	2	контроль результатов творческой деятельности
25.	Турнир по программированию. Виды конфликтов (глобальный, локальный). Роль конфликта в развитии истории.	2	1	1	наблюдение
26.	Написание короткой сцены, в которой проявляется конфликт (реализация в «Берлоге»).	2	0	2	турнир (контроль)
27.	ООП в «Берлоге». Введение в ООП. Создание объектов с методами и свойствами.	2	1	1	контроль результатов творческой деятельности
28.	Реализация простой системы объектов с параметрами и действиями.	2	0	2	контроль результатов творческой деятельности
29.	Основы 3D-моделирования. Введение в Blender: интерфейс, навигация, базовые примитивы.	2	1	1	беседа, наблюдение

30.	Моделирование простых объектов (куб, сфера, цилиндр). Трансформация (перемещение, вращение, масштаб).	2	0	2	контроль результатов творческой деятельности
31.	Режимы редактирования: вершины, рёбра, грани. Экструдирование.	2	0	2	контроль результатов творческой деятельности
32.	Создание низкополигональной модели (например, дерево, камень или простой персонаж).	2	0	2	контроль результатов творческой деятельности
33.	Наложение материалов и простых текстур в Blender.	2	1	1	контроль результатов творческой деятельности
34.	Интеграция 3D-моделей в игровой проект «Берлога» (форматы экспорта, импорт).	2	0	2	контроль результатов творческой деятельности
35.	Сквозной проект: создание собственной сцены/уровня в «Берлоге» с использованием 3D-моделей.	2	0	2	наблюдение
36.	Защита проекта (презентация сцены, демонстрация логики состояний и событий). Рефлексия.	2	1	1	защита проекта
Модуль 3. Разработка с использованием искусственного интеллекта в среде Godot Engine – 36 часов					
37.	Введение в Godot Engine. Интерфейс, сцены, узлы (Node).	2	2	0	беседа, наблюдение
38.	GDScript: переменные, функции, сигналы. Создание первой сцены.	2	1	1	контроль результатов творческой деятельности
39.	ИИ как помощник: обзор инструментов (ChatGPT, Copilot, Claude). Промпт для генерации кода.	2	1	1	беседа
40.	Генерация простых скриптов на GDScript через ИИ. Движение персонажа.	2	0	2	контроль результатов творческой деятельности

41.	ИИ для отладки: объяснение ошибок, поиск багов. Рефакторинг с помощью ИИ.	2	1	1	контроль результатов творческой деятельности
42.	Генерация игровых объектов с помощью ИИ: шаблоны узлов, экспорт переменных.	2	0	2	контроль результатов творческой деятельности
43.	ИИ для генерации диалогов и текстовых квестов. Вставка в Godot.	2	1	1	контроль результатов творческой деятельности
44.	Создание интерфейса (UI) в Godot. ИИ-генерация кода для кнопок, панелей, анимаций UI.	2	0	2	контроль результатов творческой деятельности
45.	ИИ для оптимизации кода: анализ сложности, предложения по улучшению.	2	1	1	контроль результатов творческой деятельности
46.	Генерация описаний предметов, способностей, внутриигровых подсказок через ИИ.	2	0	2	контроль результатов творческой деятельности
47.	ИИ как архитектурный советник: проектирование сцен, структуры проекта.	2	1	1	контроль результатов творческой деятельности
48.	Разработка прототипа простой 2D-игры (платформер или карточная коллекционная стратегия) с помощью ИИ.	2	0	2	наблюдение
49.	Итеративная разработка: запрос к ИИ на добавление функционала (напр. прыжки, сбор монет и пр.).	2	0	2	контроль результатов творческой деятельности
50.	ИИ для генерации тестовых сценариев и документации (README, комментарии).	2	1	1	контроль результатов творческой деятельности
51.	Сравнение кода, написанного самостоятельно и сгенерированного ИИ. Риски и ограничения.	2	1	1	беседа

52.	Работа над финальным проектом (своя мини-игра на Godot). ИИ как помощник на всех этапах.	2	0	2	наблюдение
53.	Презентация финального проекта. Демонстрация: какие части сделаны с помощью ИИ.	2	0	2	контроль результатов творческой деятельности
54.	Рефлексия: этика использования ИИ, авторство, перспективы. Защита проекта.	2	1	1	защита проекта
Модуль 4. Web-разработка с использованием искусственного интеллекта - 36 часов					
№ урока	Наименование темы	Всего	Теория	Практика	Формы аттестации
55.	Введение в веб-разработку. Клиент-сервер. HTML, CSS, JS – базовое повторение. ИИ-помощники написания кода	2	2	0	беседа, наблюдение
56.	Основы Flask: маршруты, рендеринг шаблонов, запуск локального сервера.	2	1	1	контроль результатов
57.	Генерация серверного кода на Python с помощью ИИ (ChatGPT/Copilot). Создание простого веб-приложения.	2	0	2	контроль результатов
58.	Передача данных: GET/POST параметры. Формы и обработка данных на сервере. ИИ для написания валидации.	2	0	2	контроль результатов
59.	Сессии и состояние пользователя. Генерация логики входа/регистрации через ИИ.	2	1	1	контроль результатов
60.	Шаблонизатор Jinja2: динамические страницы, циклы, условия. ИИ-генерация шаблонов.	2	1	1	контроль результатов
61.	Работа с базой данных (SQLite через Flask-SQLAlchemy). ИИ для создания моделей и запросов.	2	1	1	контроль результатов
62.	Проектирование веб-игры: идея, игровая механика на сервере (например, текстовый квест или кликер). ИИ для генерации описаний.	2	1	1	наблюдение
63.	Реализация игровой логики на Python (сервер) + простой интерфейс на HTML/CSS. ИИ как помощник в отладке.	2	0	2	контроль результатов

64.	Добавление JavaScript на фронтенд: динамические обновления без перезагрузки (fetch API). ИИ для генерации fetch-запросов.	2	0	2	контроль результатов
65.	Стилизация и адаптив: CSS + ИИ. Анимации интерфейса.	2	0	2	контроль результатов
66.	Подготовка к публикации: переменные окружения, requirements.txt, статические файлы. ИИ для генерации инструкций.	2	1	1	контроль результатов
67.	Публикация на PythonAnywhere (или аналогичном хостинге). Деплой веб-игры. ИИ для решения проблем при публикации.	2	0	2	контроль результатов
68.	Тестирование опубликованной игры. Сбор обратной связи. Итеративные улучшения с помощью ИИ.	2	0	2	наблюдение
69.	Защита проекта: демонстрация работающей онлайн-игры, презентация использования ИИ на всех этапах. Рефлексия.	2	0	2	защита проекта
70.	Обобщение тем курса. Обсуждение вектора дальнейшего развития	2	2	0	беседа, наблюдение
71.	Подготовка к итоговой выставке.	2	1	1	контроль результатов
72.	Проведение итоговой выставки проектов	2	0	2	защита проекта
	Итого:	**144**	**41**	**103**	

Содержание учебного плана
дополнительной общеразвивающей программы
«IT-квантум. Введение в компьютерную разработку»
2 год обучения

Тема 1. Введение в командную разработку. Жизненный цикл IT-проекта – 36 часов
Теория: Понятие командной разработки, её отличие от индивидуальной. Преимущества и вызовы работы в команде. Жизненный цикл программного продукта: анализ → проектирование → разработка → тестирование → внедрение → сопровождение. Модели жизненного цикла: каскадная, итеративная, Agile (обзор).
Практика: Дискуссия «Почему большие проекты невозможно создать в одиночку?». Составление схемы жизненного цикла для вымышленного IT-проекта (например, мобильного приложения).
Форма контроля: Устный опрос, проверка схемы.

Тема 2. Роли в команде: продюсер, PM, разработчик, тестировщик, дизайнер. Распределение ролей
Теория: Описание ключевых ролей в IT-команде: продюсер (видение продукта), проектный менеджер (процессы), разработчик (код), тестировщик (качество), дизайнер (интерфейс и опыт). Дополнительные роли: аналитик, DevOps, документатор. Ответственность и зоны взаимодействия.
Практика: Упражнение «Собери команду»: для заданного проекта (например, создание визуальной новеллы) участники распределяют роли внутри мини-групп, аргументируют выбор. Составление матрицы ответственности.
Форма контроля: Контроль результатов творческой деятельности (схема распределения ролей).

Тема 3. Системы управления проектами. Создание доски задач
Теория: Назначение систем управления проектами. Обзор «Битрикс24» (карточки, списки, метки), использование совместных электронных таблиц для планирования. Выбор инструмента под размер команды. Базовые элементы: задачи (карточки), статусы, исполнители, дедлайны.
Практика: Регистрация в выбранной системе. Создание доски для учебного проекта: колонки «To do», «In progress», «Done». Добавление 3–4 задач, назначение ответственных.
Форма контроля: Проверка созданной доски, скриншот или ссылка.

Тема 4. Командная работа: мозговой штурм, определение идеи сквозного проекта Сервис Яндекс.Доски
Теория: Методы генерации идей: мозговой штурм (правила, фазы), метод SCAMPER, майндмэппинг. Критерии выбора идеи для сквозного проекта: реализуемость за спринты, наличие технического задела, интерес команды.
Практика: Проведение мозгового штурма в командах (3–4 человека), фиксация всех идей на доске (электроной или физической). Выбор одной идеи голосованием. Формулировка названия и краткого описания проекта.
Форма контроля: Наблюдение, утверждение идеи наставником.

Тема 5. Agile и Scrum: спринты, бэклог, ежедневные встречи

Теория: Принципы Agile (гибкость, итеративность, обратная связь). Scrum: роли (Scrum-мастер, владелец продукта, команда), артефакты (бэклог, спринт-бэклог, инкремент), события (планирование спринта, ежедневный стендап, ревью, ретроспектива). Длительность спринта (1–2 недели). Бэклог продукта – список всех желаемых функций. Приоритизация: метод MoSCoW, матрица усиление/ценность. Планирование спринта: отбор задач из бэклога, разбиение на подзадачи, оценка трудоёмкости (story points или часы).
Практика: Моделирование спринта: команды формируют недельный бэклог из 5–7 задач, выбирают задачи на спринт (2 дня или 1 неделя). Задачи daily standup. Проведение короткого стендапа (имитация). Асинхронный daily standup. Фиксирование итогов коротким сообщением в командном чате. Использование шаблонов или стикеров для заполнения информации заранее

Форма контроля: Контроль результатов творческой деятельности (бэклог и план спринта).

Тема 6. Git: установка, базовые команды (init, add, commit, status, log). Ветки (branch, checkout, merge), разрешение конфликтов
Теория: Система контроля версий Git: назначение, основные понятия (репозиторий, коммит, ветка). Установка Git, настройка имени пользователя. Команды: git init, git add, git commit, git status, git log. Ветвление: git branch, git checkout, git merge. Понятие конфликта и его разрешение (ручное редактирование).
Практика: Установка Git (если не установлен). Создание локального репозитория, добавление файла, коммит. Создание ветки, переключение, слияние. Искусственное создание конфликта и его разрешение.

Форма контроля: Контроль результатов творческой деятельности (скриншоты выполненных команд, репозиторий на локальной машине).

Тема 7. Разработка спринта 1: распределение задач, работа в ветках Git. Ежедневные стендапы (имитация + дистант). Ревью кода внутри команды
Теория: Как организовать работу в спринте: назначение исполнителей, создание веток для каждой задачи, регулярные коммиты. Ежедневный стендап (что сделано, что планирую, какие блокеры). Ревью кода: цели, чек-лист, конструктивная обратная связь.
Практика: Команды выполняют задачи первого спринта (программирование, вёрстка, дизайн – в зависимости от проекта). Проводят 2–3 коротких стендапа (один – очно, остальные – в чате). Организуют ревью кода через Pull request.
Форма контроля: Контроль результатов творческой деятельности (выполненные задачи, активность в репозитории, участие в ревью).

Тема 8. Работа с удалённым репозиторием: Pull request и code review. Баг-трекинг: оформление issue, метки, приоритеты
Теория: Удалённые репозитории (GitVerse, Mos.hub). Команды git remote, git push, git pull, git clone. Процесс Pull request: форк, ветка, запрос на слияние, обсуждение, ревью кода. Баг-трекинг: задачи (issues), метки (bug, enhancement, question), приоритеты (high, medium, low).
Практика: Регистрация на GitVerse, создание удалённого репозитория. Отправка локальных коммитов, клонирование. Создание ветки на GitVerse, открытие Pull request. Добавление issue к репозиторию с меткой и приоритетом.
Форма контроля: Контроль результатов творческой деятельности (ссылка на репозиторий, активный PR и issue).

Тема 9. Разработка спринта 2: добавление функционала, решение конфликтов. Тестирование и приёмка: оформление багов, приоритизация
Теория: Продолжение работы: добавление новых функций. Конфликты при слиянии (git merge) – причины и способы разрешения. Тестирование: виды тестов (модульное, интеграционное, ручное). Оформление багов: воспроизведение, ожидаемое/фактическое поведение, серьёзность и приоритет.
Практика: Команды выполняют задачи второго спринта. При возникновении конфликтов в Git – разрешают их. Проводят ручное тестирование друг у друга (cross-testing). Оформляют баги в issue с метками и приоритетами.
Форма контроля: Контроль результатов творческой деятельности (завершённые задачи спринта, исправленные баги, разрешённые конфликты).

Тема 10. CI/CD: знакомство (GitVerse). Автоматическая сборка и тесты

Теория: Понятие CI/CD (непрерывная интеграция и непрерывная доставка). Зачем автоматизировать сборку, тестирование, деплой. Примеры: GitHub Actions, GitLab CI, Jenkins. Базовые понятия: workflow, job, step, runner, конфигурационный файл (YAML).
Практика: Создание простого workflow в GitVerse для своего репозитория: запуск проверки синтаксиса (например, для Python или Markdown) при каждом push. Просмотр результатов в интерфейсе GitHub.
Форма контроля: Контроль результатов творческой деятельности

Тема 11. Разработка спринта 3: доработка по фидбеку. Интеграционное тестирование и финальная сборка
Теория: Учёт обратной связи: приоритизация изменений и добавление в бэклог. Интеграционное тестирование – проверка взаимодействия всех компонентов. Финальная сборка: компиляция, создание исполняемого файла, подготовка дистрибутива или деплой на

хостинг.
Практика: Команды вносят изменения, запланированные на спринт 3 (по обратной связи). Проводят интеграционное тестирование (прогоняют сценарий от начала до конца). Выполняют финальную сборку своего сквозного проекта (если игра – экспорт, если веб-приложение – деплой).
Форма контроля: Контроль результатов творческой деятельности (работоспособная финальная сборка, отчёт о тестировании).

Тема 12. Техническая документация: README, Wiki, архитектурные схемы
Теория: Зачем нужна документация. Файл README (назначение, структура: название, описание, установка, использование, лицензия). Wiki как расширенная документация. Архитектурные схемы: диаграммы потоков данных, диаграммы классов, компонентные диаграммы. Инструмент - draw.io.
Практика: Создание README для учебного проекта в формате Markdown. Построение простой архитектурной схемы (например, «пользователь → веб-интерфейс → сервер → база данных») в draw.io.
Форма контроля: Контроль результатов творческой деятельности (файл README в репозитории, схема в формате изображения).

Тема 13. Демо прототипа. Сбор обратной связи. Ретроспектива
Теория: Цель демо-встречи (спринт-ревью) – показать результат заинтересованным лицам. Как собирать обратную связь (опросы, устные комментарии, фиксация пожеланий). Ретроспектива: анализ того, что пошло хорошо, что плохо, что улучшить. Техники «Start, Stop, Continue» или «Море, якоря, облака».
Практика: демонстрация работающего продукта (промежуточный результат) Получение обратной связи. Проведение ретроспективы внутри команды с заполнением таблицы «Что улучшить».
Форма контроля: Защита этапа проекта (демо спринта), ретроспективная карта.

Тема 14. Внесение изменений в проект на основании обратной связи. Рефакторинг кода. Код-ревью. Обсуждение Pull Request в GitVerse. Pinning tests
Теория: Обратная связь как источник улучшений: сбор, систематизация, приоритизация (критические, желательные). Рефакторинг – изменение внутренней структуры без изменения поведения (устранение дублирования, длинных функций, магических чисел). Код-ревью: цели, чек-лист ревьювера, правила конструктивной обратной связи. Pull Request в GitVerse: создание, описание, запрос ревью, обсуждение комментариев, доработка, слияние. Pinning tests (тесты на регрессию) – фиксирующие тесты, которые проверяют, что исправленная ошибка не вернётся (сначала тест падает, после исправления – проходит).
Практика: Команды получают обратную связь по сквозному проекту, составляют список изменений. Каждый участник выполняет рефакторинг одного фрагмента кода. В парах проводят код-ревью: один создаёт Pull Request с изменениями, другой оставляет комментарии, автор дорабатывает и мержит. Для исправленного бага пишут тест (в CI или простой assert), демонстрируя, что он падает до правки и проходит после.
Форма контроля: Контроль результатов творческой деятельности (оформленный Pull Request с ревью и мержем, добавленный тест, отчёт о рефакторинге).

Тема 15: AI-помощь в модификации кода
Теория: AI-анализ кода: выявление антипаттернов, мёртвого кода, логических ошибок. Комментирование: фиксация намерения, инвариантов, предусловий. Точки модификации по ТЗ: семантическое сопоставление требований с модулями, классификация (добавить/изменить/удалить). Генерация изменений: фрагменты кода с обоснованием выбора алгоритма. Проверка: синтаксис (AST, линтеры), логика (граничные случаи), генерация тестов.

Рефакторинг: единый стиль, устранение дублирования, переименование. Отчёт: журнал правок, обновлённая документация, риски.
Практика: Анализ текущей кодовой базы при помощи **AI**, комментирование, и выявление точек модификации согласно техническому заданию; генерация конкретных изменений (фрагменты кода, правки структур данных, оптимизация алгоритмов) с пояснением логики; проверка изменений на синтаксические и логические ошибки, предложение тестовых сценариев; рефакторинг и унификация стиля для поддержки сопровождаемости; документирование внесённых правок и формирование итогового отчёта. .
Форма контроля: Контроль результатов творческой деятельности (оформленный Pull Request с ревью и мержем, добавленный тест, отчёт о рефакторинге).

Тема 16. Презентация готового продукта
Теория: Подготовка к презентации: структура (проблема, решение, демонстрация, технологии, дальнейшие планы). Правила публичного выступления: контакт с аудиторией, управление временем, ответы на вопросы. Роль каждого члена команды в презентации.
Практика: Репетиция презентации в команде. Публичная презентация. Ответы на вопросы.
Форма контроля: Защита проекта

Тема 17. Итоговая аттестация по модулю: защита портфолио (доска задач, репозиторий, документация)
Теория: Что входит в портфолио: ссылка на репозиторий **GitVerse**, доска задач, README, пользовательская документация, архитектурные схемы, отчёты о тестировании, ретроспективные карты. Критерии оценки полноты и качества портфолио.
Практика: Оформление портфолио в едином документе. Индивидуальное или командное собеседование с наставником: защита портфолио, обоснование принятых решений.
Форма контроля: Защита портфолио (оценка наставника, самооценка команды).

Тема 18. Рефлексия образовательных результатов
Теория: Значение рефлексии в проектном обучении. Вопросы для самоанализа: какие навыки командной работы приобрели, что было самым сложным, какой вклад внёс каждый участник, как изменилось понимание IT-проектов. Оценка своей роли и достижений.
Практика: Заполнение индивидуального рефлексивного листа (например, по технике «Плюс-минус-интересно»). Групповое обсуждение: обмен впечатлениями о модуле, советы будущим командам.
Форма контроля: Беседа, наблюдение, письменная рефлексия.

Модуль 2 второго года обучения: «Событийное программирование в среде “Берлога” + основы 3D-моделирования»

Тема 19. Знакомство с игрой «Берлога: Защита пасеки». Разбор структуры готового игрового проекта
Теория: Обзор среды «Берлога»: назначение, интерфейс, основные возможности. Запуск готового проекта «Защита пасеки». Определение ключевых игровых объектов (медведь, пчёлы, улей, пасечник). Общее представление о логике: что происходит при касании, как начисляются очки, условия проигрыша/выигрыша.
Практика: Запуск проекта, игра в режиме тестирования. Исследование структуры проекта: список объектов, настройки сцены, блоки поведения. Заполнение таблицы «Объект – его роль – пример действия».
Форма контроля: Беседа, наблюдение; таблица описания объектов.

Тема 20. Изучение логики поведения объектов. Создание схемы игрового процесса с выделением ключевых элементов логики

Теория: Понятие «логика поведения»: как объекты реагируют на действия игрока и друг на друга. Составление блок-схемы игрового процесса (старт → движение → столкновение → изменение счётчика → окончание).
Практика: Разбор логики двух-трёх объектов в «Защите пасеки». Создание графической схемы игрового процесса с выделением условий и переходов. Презентация схемы в группе.
Форма контроля: Контроль результатов творческой деятельности (схема игрового процесса).

Тема 21. Машины состояний. Понятие состояния. Создание системы с простыми состояниями
Теория: Что такое состояние объекта (например, «идёт», «стоит», «атакует»). Конечный автомат (Finite State Machine) как модель поведения. Примеры из жизни: светофор, лифт. Зачем нужны состояния в играх (упрощение логики, наглядность).
Практика: В среде «Берлога» создание двух состояний для простого объекта (например, для пчелы: «летает» и «отдыхает»). Реализация переключения по таймеру. Проверка работы.
Форма контроля: Контроль результатов творческой деятельности (проект с двумя состояниями).

Тема 22. Машины состояний. Переходы. Создание схемы машины состояний для персонажа
Теория: Понятие перехода (транзиции). Условия перехода (по времени, по событию, по значению переменной). Способы задания переходов в «Берлоге» (блоки «если», события столкновения и т.п.).
Практика: Выбор персонажа (медведь). Проектирование схемы его состояний: «спит», «идёт к улью», «атакует», «убегает». Прорисовка переходов между состояниями. Реализация двух переходов в проекте (например, по таймеру и по касанию).
Форма контроля: Контроль результатов творческой деятельности (схема + фрагмент проекта).

Тема 23. Машины состояний. События. Обработка событий в игровой логике. Роль триггеров и условий
Теория: События (клик, касание, столкновение, получение сообщения). Триггеры как активаторы переходов. Условия (проверка переменных). Разница между событием и условием: событие происходит (единоразово), условие может длиться.
Практика: Добавление в проект события «при касании медведя с ульем» – переход в состояние «атака». Добавление условия «если здоровье улья < 0» – переход в состояние «разрушен». Реализация в «Берлоге».
Форма контроля: Контроль результатов творческой деятельности (работающая реакция на события).

Тема 24. Реализация логики смены состояний на практике (в среде «Берлога»)
Теория: Повторение и обобщение: цикл состояний, обработка ошибок (зависание в состоянии). Рекомендации по именованию состояний и переменных.
Практика: Командная работа: каждый участник создаёт мини-сцену с двумя персонажами (например, кот и мышь), у каждого не менее трёх состояний с переходами по событиям. Взаимное тестирование (один играет, другой смотрит на корректность смены состояний).
Форма контроля: Контроль результатов творческой деятельности (проект с полной машиной состояний).

Тема 25. Турнир по программированию. Виды конфликтов (глобальный, локальный). Роль конфликта в развитии истории
Теория: Что такое конфликт в нарративе и игровом дизайне. Глобальный конфликт (война, вторжение, катастрофа) и локальный (спор персонажей, внутренняя борьба). Роль конфликта в мотивации игрока и смене состояний.
Практика: Дискуссия «Примеры конфликтов в известных играх». Работа в парах: придумать

по одному глобальному и локальному конфликту, которые можно реализовать в «Берлоге» (например, защита пасеки – глобальный; конкуренция двух пчёл за мёд – локальный).
Форма контроля: Наблюдение, устные ответы.

Тема 26. Написание короткой сцены, в которой проявляется конфликт (реализация в «Берлоге»)
Теория: Как перенести конфликт в игровую механику: конфликт как условие смены состояний, как цель уровня. Структура короткой сцены (завязка, обострение, разрешение).
Практика: Участники проектируют и реализуют в «Берлоге» короткую сцену (1–2 минуты геймплея), где явно показан конфликт. Сцена должна использовать машины состояний (минимум два персонажа с состояниями). Проведение мини-турнира: сравнение работ, голосование за лучшую реализацию конфликта.
Форма контроля: Соревнование. Турнир

Тема 27. ООП в «Берлоге». Введение в ООП. Создание объектов с методами и свойствами
Теория: Понятие объектно-ориентированного программирования (объекты, свойства, методы). Как ООП применяется в играх. В «Берлоге» каждый спрайт – это объект, можно задавать его переменные (свойства) и блоки действий (методы).
Практика: Создание пользовательского объекта «Игровой персонаж»: добавление свойств «сила», «скорость», «здоровье». Создание метода «атаковать», который уменьшает здоровье цели. Демонстрация вызова метода по событию.
Форма контроля: Контроль результатов творческой деятельности (проект с объектом, имеющим свойства и методы).

Тема 28. Реализация простой системы объектов с параметрами и действиями
Теория: Инкапсуляция: объект сам отвечает за свои данные. Обмен сообщениями между объектами. Проектирование системы: например, несколько врагов с разными параметрами (здоровье, урон) и единый метод «получить урон».
Практика: Разработка в «Берлоге» системы из 2–3 объектов (игрок, враг, бонус). У каждого объекта – свои параметры (здоровье, сила). Реализация взаимодействия: при касании враг атакует игрока, игрок теряет здоровье; при касании бонуса здоровье восстанавливается. Использование методов объектов.
Форма контроля: Контроль результатов творческой деятельности (рабочая система объектов).

Тема 29. Основы 3D-моделирования. Введение в Blender: интерфейс, навигация, базовые примитивы
Теория: Понятие 3D-модели. Области применения (игры, анимация, визуализация). Обзор интерфейса Blender: 3D-окно, панель инструментов, свойства объектов. Навигация (вращение, панорамирование, зуминг). Добавление примитивов (куб, сфера, цилиндр).
Практика: Установка Blender (если не установлен). Запуск, знакомство с интерфейсом. Создание сцены с тремя примитивами (куб, сфера, цилиндр), их перемещение, вращение, масштабирование. Сохранение проекта.
Форма контроля: Беседа, наблюдение; проверка сохранённой сцены.

Тема 30. Моделирование простых объектов (куб, сфера, цилиндр). Трансформация (перемещение, вращение, масштаб)
Теория: Повторение инструментов трансформации (горячие клавиши: G – перемещение, R – вращение, S – масштаб). Применение трансформаций к примитивам для создания простых форм (например, вытянутый куб – кирпич, сплюснутая сфера – блин).
Практика: Моделирование трёх простых объектов из примитивов: стол (куб + 4 цилиндра), дерево (цилиндр + сфера), камень (искажённая сфера). Экспорт в формате .obj или .fbx.
Форма контроля: Контроль результатов творческой деятельности (файлы моделей).

Тема 31. Режимы редактирования: вершины, рёбра, грани. Экструдирование
Теория: Режим Edit Mode (Tab). Выделение вершин/рёбер/граней. Инструменты: перемещение, вращение, масштабирование выделенного. Экструдирование (E) – создание новой геометрии из выделенной грани. Применение: создание углублений, выступов.
Практика: На основе куба создание простого домика (экструдирование крыши, выдавливание окна). Сохранение промежуточных результатов. Практика с инструментом Loop Cut (разрезание) для добавления деталей.
Форма контроля: Контроль результатов творческой деятельности (модель домика в формате .blend).

Тема 32. Создание низкополигональной модели (например, дерево, камень или простой персонаж)
Теория: Понятие low-poly (малое количество полигонов), преимущества для игр (производительность, стиль). Техники: использование примитивов, экструдирование без излишней детализации.
Практика: По выбору: создание низкополигонального дерева (ствол из цилиндра, крона из нескольких кубов), камня (искажённый куб с экструдированными гранями) или простого персонажа (кубический робот). Соблюдение количества полигонов (не более 500).
Форма контроля: Контроль результатов творческой деятельности (готовая low-poly модель).

Тема 33. Наложение материалов и простых текстур в Blender
Теория: Понятие материала (цвет, блеск, прозрачность). Текстура – изображение, накладываемое на модель. Режим Shading Editor: создание материала, изменение базового цвета, добавление шума или изображения. UV-развёртка (базовое понятие).
Практика: Присвоение цвета созданной low-poly модели (дерево – коричневый ствол, зелёная крона). Добавление простой текстуры (например, растрового изображения травы) для земли. Настройка отражения (glossy) для металлических деталей.
Форма контроля: Контроль результатов творческой деятельности (модель с материалом, скриншот).

Тема 34. Интеграция 3D-моделей в игровой проект «Берлога» (форматы экспорта, импорт)
Теория: Поддерживаемые форматы в «Берлоге» (обычно .obj, .fbx, .dae). Экспорт из Blender: настройки (масштаб, поворот осей). Импорт в «Берлогу»: добавление модели как нового объекта, настройка коллизии, масштабирование.
Практика: Экспорт созданной модели в формат .obj. Импорт модели в проект «Берлога» (например, замена стандартного медведя на своего персонажа). Настройка физических свойств (вес, упругость). Тестирование в сцене.
Форма контроля: Контроль результатов творческой деятельности (работающая сцена с импортированной моделью).

Тема 35. Сквозной проект: создание собственной сцены/уровня в «Берлоге» с использованием 3D-моделей
Теория: Этапы создания уровня: концепт (что это за место), расстановка объектов, настройка логики состояний и событий, тестирование баланса. Требования к проекту: не менее 3 объектов, собранных в Blender, использование машин состояний для хотя бы двух персонажей, наличие конфликта (цели).
Практика: Разработка сцены в командах (или индивидуально). Создание трёх моделей в Blender, импорт, расстановка на сцене. Программирование поведения с использованием состояний и событий. Тестирование, отладка.
Форма контроля: Наблюдение, промежуточные консультации.

Тема 36. Защита проекта (презентация сцены, демонстрация логики состояний и событий). Рефлексия
Теория: Структура презентации проекта: название, цель, используемые модели, схема машины состояний, демонстрация геймплея, что получилось, что планируется улучшить.
Практика: Публичная защита созданного уровня перед наставником и другими учащимися (7–10 минут). Демонстрация работы состояний, событий, импортированных 3D-моделей. Вопросы от аудитории. Заполнение индивидуального рефлексивного листа (что освоили, что было сложным).
Форма контроля: Защита проекта (оценка по критериям: завершённость, работа логики, использование 3D, качество презентации), рефлексия.

Модуль 3 второго года обучения:
«Разработка с использованием искусственного интеллекта в среде Godot Engine» - 36 часов

Тема 37. Введение в Godot Engine. Интерфейс, сцены, узлы (Node)
Теория: Обзор Godot Engine: свободный игровой движок, поддержка 2D и 3D, сценарная система. Понятие сцены (Scene) и узла (Node). Иерархия узлов. Основные типы узлов: Node2D, Sprite, CollisionShape, Area2D. Интерфейс редактора (2D/3D/скрипт).
Практика: Установка Godot (последней стабильной версии). Создание нового проекта. Добавление узла Sprite, загрузка текстуры-заглушки. Запуск сцены. Сохранение проекта.
Форма контроля: Беседа, наблюдение; проверка созданной сцены.

Тема 38. GDScript: переменные, функции, сигналы. Создание первой сцены
Теория: Введение в GDScript (синтаксис, отличие от Python). Переменные (var, const), типы (int, float, String, bool). Функции (func), параметры, возвращаемые значения. Сигналы (signal) – механизм событий. Подключение сигнала к функции через редактор или код.
Практика: Создание сцены с узлом KinematicBody2D (игрок). Написание скрипта для движения по стрелкам (переменные скорости, функция _process). Подключение сигнала body_entered для обнаружения столкновений. Запуск и тестирование.
Форма контроля: Контроль результатов творческой деятельности (работающая сцена с движением).

Тема 39. ИИ как помощник: обзор инструментов (ChatGPT, Copilot, Claude). Промпт для генерации кода
Теория: Современные ИИ-инструменты для разработчиков: ChatGPT (веб-интерфейс, API), GitHub Copilot (плагин для IDE), Claude (анализ больших объёмов кода). Что такое промпт (запрос): структура, контекст, примеры. Рекомендации по формулировке: указать язык, версию движка, желаемый функционал, ожидаемый формат вывода.
Практика: Регистрация/доступ к одному из инструментов (например, ChatGPT). Составление промпта для генерации кода движения персонажа в Godot на GDScript. Сравнение сгенерированного кода с кодом из темы 2. Анализ различий.
Форма контроля: Беседа, демонстрация составленного промпта и полученного кода.

Тема 40. Генерация простых скриптов на GDScript через ИИ. Движение персонажа
Теория: Особенности генерации игровой логики: учёт физических свойств (скорость, ускорение), типы движения (клавиши, геймпад). Типичные ошибки ИИ (неправильные имена узлов, устаревший синтаксис). Проверка и адаптация сгенерированного кода.
Практика: Запрос к ИИ: «Напиши скрипт для персонажа в Godot 4 (GDScript), который движется влево/вправо, прыгает и проверяет, стоит ли на земле». Интеграция кода в проект. Настройка коллизий, проверка работоспособности.
Форма контроля: Контроль результатов творческой деятельности (рабочий скрипт движения и прыжков).

Тема 41. ИИ для отладки: объяснение ошибок, поиск багов. Рефакторинг с помощью ИИ
Теория: Как ИИ может помочь в отладке: скопировать сообщение об ошибке, запросить объяснение. Уточняющие вопросы: «Почему переменная не видна в другой функции?». Рефакторинг: запрос на упрощение кода, улучшение читаемости, выделение методов.
Практика: Внесение намеренной ошибки в код движения (например, неверное имя сигнала). Запрос к ИИ с текстом ошибки. Исправление кода по рекомендации ИИ. Запрос на рефакторинг (выделение функции прыжка в отдельный метод). Сравнение исходного и рефакторированного кода.
Форма контроля: Контроль результатов творческой деятельности (исправленный и рефакторированный код, отчёт о запросах).

Тема 42. Генерация игровых объектов с помощью ИИ: шаблоны узлов, экспорт переменных
Теория: Создание повторяемых игровых объектов (враги, монеты, бонусы). Использование `@export` для настройки параметров в редакторе. ИИ для генерации шаблона сцены (PackedScene) и скрипта.
Практика: Запрос к ИИ: «Создай скрипт для монеты в Godot 4: при касании игрока проигрывается звук, добавляется 10 очков, монета исчезает». Реализация скрипта. Создание сцены монеты как отдельного узла, экспорт переменной `score_value`. Размещение нескольких монет на уровне.
Форма контроля: Контроль результатов творческой деятельности (сцена монеты, работающая механика сбора).

Тема 43. ИИ для генерации диалогов и текстовых квестов. Вставка в Godot
Теория: Генерация текстового контента с помощью ИИ (NPC-реплики, описания квестов, внутриигровые письма). Структура диалоговой системы в Godot: узлы Control, RichTextLabel, сигналы кнопок. Интеграция сгенерированного текста.
Практика: Запрос к ИИ: «Сгенерируй 5 фраз для торговца в фэнтези-игре». Добавление в сцену простого диалогового окна (Panel + RichTextLabel + Button). Вывод случайной фразы при нажатии. Расширение: диалог с выбором ответов.
Форма контроля: Контроль результатов творческой деятельности (работающий диалог со сгенерированными текстами).

Тема 44. Создание интерфейса (UI) в Godot. ИИ-генерация кода для кнопок, панелей, анимаций UI
Теория: Элементы UI: Control, Button, Panel, Label, TextureProgressBar. Анимация UI (изменение размера, цвета, прозрачности) с помощью AnimationPlayer. ИИ для генерации кода обработки нажатий и смены сцен.
Практика: Запрос к ИИ: «Создай скрипт для главного меню в Godot: кнопки “Старт”, “Настройки”, “Выход”. При нажатии “Старт” загружается сцена уровня». Реализация меню. Добавление анимации появления кнопок (плавное увеличение и изменение цвета).
Форма контроля: Контроль результатов творческой деятельности (рабочее главное меню, переход на уровень).

Тема 45. ИИ для оптимизации кода: анализ сложности, предложения по улучшению
Теория: Понятие производительности в играх (FPS, количество объектов, циклы в `_process`). ИИ как инструмент для оценки сложности алгоритмов и выявления узких мест. Типовые рекомендации: избегать `get_node()` в каждом кадре, кэшировать ссылки.
Практика: Предоставить ИИ фрагмент кода с потенциально неоптимальным циклом (например, поиск всех монет на уровне через `get_tree().get_nodes_in_group()` в каждом кадре). Запрос «Предложи оптимизацию». Рефакторинг кода по рекомендации. Измерение производительности до и после (отображение FPS).

Форма контроля: Контроль результатов творческой деятельности (оптимизированный код, краткий отчёт о замерах).

Тема 46. Генерация описаний предметов, способностей, внутриигровых подсказок через ИИ
Теория: Использование ИИ для наполнения игры контентом (лор, описание предметов, названия способностей, подсказки для загрузочных экранов). Форматирование вывода (JSON, CSV) для удобной вставки в Godot.
Практика: Запрос к ИИ: «Сгенерируй список из 5 предметов для RPG-игры. Для каждого укажи название, описание, редкость (обычный, редкий, эпический) и эффект (+5 здоровья, +2 силы) в формате JSON». Импорт JSON в Godot (файл items.json). Создание сцены для отображения случайного предмета с использованием данных.
Форма контроля: Контроль результатов творческой деятельности (работающее отображение предметов из сгенерированного JSON).

Тема 47. ИИ как архитектурный советник: проектирование сцен, структуры проекта
Теория: Понятие архитектуры проекта: как организованы сцены, глобальные автозагрузки (singletons), сигналы для связи узлов. ИИ может предложить схему взаимодействия компонентов. Пример: «Как организовать систему здоровья игрока, которая обновляет интерфейс и вызывает загрузку сцены поражения?».
Практика: Запрос к ИИ описать структуру для простой игры (например, платформер с монетами, врагами и счетом очков). Построение схемы на основе рекомендации (узлы: Player, Enemy, UI, GameManager автозагрузка). Реализация предложенной архитектуры в проекте (хотя бы каркас).
Форма контроля: Контроль результатов творческой деятельности (схема архитектуры, реализованный каркас).

Тема 48. Разработка прототипа простой 2D-игры (платформер или карточная коллекционная стратегия) с помощью ИИ
Теория: Что такое прототип (минимальная версия с ключевой механикой). Варианты: платформер (прыжки, сбор предметов, враг) или карточная ККИ (выбор карты, атака, изменение очков здоровья). Роль ИИ на этапе прототипирования: генерация базового кода, ускорение итераций.
Практика: Команды выбирают жанр. С помощью ИИ генерируют: для платформера – скрипты движения, прыжков, сбора монет; для ККИ – скрипты выбора карт, подсчёта урона. Интеграция в Godot, получение первого играбельного прототипа (без финальной графики).
Форма контроля: Наблюдение, демонстрация прототипа (к концу занятия).

Тема 49. Итеративная разработка: запрос к ИИ на добавление функционала (напр. прыжки, сбор монет и пр.)
Теория: Итеративный процесс: начать с минимума, затем запрашивать у ИИ код для новой функции. Как правильно формулировать запрос в контексте существующего кода (предоставлять фрагменты, указывать имена узлов). Управление сложностью.
Практика: На основе прототипа из темы 12 команды формулируют задачу на добавление одной новой функции (например, «добавить врага, который движется влево-вправо» или «добавить анимацию выбора карты»). Запрос к ИИ, интеграция, тестирование. Документирование изменений.
Форма контроля: Контроль результатов творческой деятельности

Тема 50. ИИ для генерации тестовых сценариев и документации (README, комментарии)
Теория: Зачем нужны тестовые сценарии (ручное тестирование: "пройти уровень без сбора монет", "столкнуться с врагом трижды"). ИИ может сгенерировать список чек-листов.

Документация: комментарии в коде, README для репозитория.
Практика: Запрос к ИИ: «Сгенерируй 5 тестовых сценариев для платформера с монетами и врагом». Запрос: «Напиши README для игры [название] на Godot: описание, управление, установка». Добавление комментариев к сложным функциям с помощью ИИ (запрос «прокомментируй этот код»).
Форма контроля: Контроль результатов творческой деятельности

Тема 51. Сравнение кода, написанного самостоятельно и сгенерированного ИИ. Риски и ограничения
Теория: Обсуждение различий: стиль, эффективность, избыточность, безопасность. Риски: слепое копирование кода, неподходящие архитектурные решения, генерация неоптимальных или устаревших конструкций. Ограничения ИИ (отсутствие понимания контекста всего проекта).
Практика: Рефакторинг кода, сгенерированных ИИ ранее для улучшения или адаптации. Сравнение двух версий (читаемость, количество строк, производительность). Групповое обсуждение
Форма контроля: Беседа, письменные выводы

Тема 52. Работа над финальным проектом (своя мини-игра на Godot). ИИ как помощник на всех этапах
Теория: Этапы разработки финального проекта: концепт, прототип, полировка, тестирование. Роль ИИ на каждом этапе: генерация идей, кода, диалогов, UI, документации, тестов. Самостоятельная организация работы.
Практика: Команды (или индивидуальные участники) разрабатывают мини-игру (продолжительность геймплея 2–5 минут) с использованием Godot и ИИ-инструментов. Допускается использование всех изученных приёмов. Наставник проводит промежуточные консультации, помогает сформулировать промпты.
Форма контроля: Наблюдение, промежуточные отчёты

Тема 53. Презентация финального проекта. Демонстрация: какие части сделаны с помощью ИИ
Теория: Структура презентации: название, жанр, основные механики, демонстрация геймплея, перечень использованных ИИ-инструментов, конкретные примеры промптов и сгенерированных фрагментов. Оценка вклада ИИ и собственного вклада.
Практика: Публичная презентация мини-игры перед наставником и другими учащимися (5–7 минут). Демонстрация работы игры. Показ исходного кода, где видно использование ИИ (комментарии или отдельный слайд). Ответы на вопросы.
Форма контроля: Контроль результатов творческой деятельности (оценка игры и презентации).

Тема 54. Рефлексия: этика использования ИИ, авторство, перспективы. Защита проекта
Теория: Этические аспекты: плагиат, авторство сгенерированного кода, указание использования ИИ в документации. Перспективы развития. Защита проекта как итоговая аттестация.
Практика: Заполнение рефлексивного листа (что дало использование ИИ, какие были трудности, как изменилось отношение к программированию). Групповая дискуссия «Должен ли разработчик уметь писать код без ИИ?». Финальная защита проекта с выставлением оценки (критерии: завершённость, стабильность, использование ИИ, понимание кода).
Форма контроля: Защита проекта, беседа, письменная рефлексия.

Модуль 4 второго года обучения: «Веб-разработка с использованием ИИ (Flask, фронтенд, деплой)»

Тема 55. Введение в веб-разработку. Клиент-сервер. HTML, CSS, JS – базовое повторение. ИИ-помощники написания кода
Теория: Архитектура клиент-сервер: роль браузера (клиент) и веб-сервера. Основы HTTP (запрос, ответ, методы GET/POST). Повторение HTML (структура страницы, формы), CSS (цвета, шрифты, позиционирование), JavaScript (переменные, функции, обработка событий). Обзор ИИ-инструментов для веб-разработки (ChatGPT, Copilot, Codeium).
Практика: Написание простой HTML-страницы с формой и кнопкой. Добавление CSS-стилей. Написание JS-обработчика нажатия кнопки (вывод сообщения). Запрос к ИИ: «Создай форму логина на HTML/CSS с полями username и password». Анализ сгенерированного кода.
Форма контроля: Беседа, наблюдение; проверка созданной страницы.

Тема 56. Основы Flask: маршруты, рендеринг шаблонов, запуск локального сервера
Теория: Введение в Flask (микрофреймворк для Python). Установка Flask через pip. Маршруты (декоратор @app.route). Рендеринг HTML-шаблонов с помощью render_template. Запуск сервера (app.run(debug=True)). Структура проекта (папки templates/, static/).
Практика: Установка Flask, создание минимального приложения с двумя маршрутами (/ и /about). Создание шаблона index.html с переменной (например, приветствие по имени). Запуск сервера, проверка в браузере.
Форма контроля: Контроль результатов творческой деятельности (работающее Flask-приложение).

Тема 57. Генерация серверного кода на Python с помощью ИИ (ChatGPT/Copilot). Создание простого веб-приложения
Теория: Особенности генерации кода для Flask: указание версии Python, описание маршрутов, шаблонов. Как проверять и адаптировать сгенерированный код (синтаксис, имена файлов).
Практика: Запрос к ИИ для создания сервера игры. Интеграция кода в проект. Запуск, тестирование. Расширение: добавить статический CSS-файл.
Форма контроля: Контроль результатов творческой деятельности (работающее приложение, сгенерированный код адаптирован).

Тема 58. Передача данных: GET/POST параметры. Формы и обработка данных на сервере. ИИ для написания валидации
Теория: Различие GET (параметры в URL) и POST (в теле запроса). Доступ к GET-параметрам через request.args, к POST – через request.form. Обработка отправки формы. Валидация данных (проверка на пустоту, формат email и т.д.).
Практика: Создание формы регистрации (имя, email, пароль). Обработчик формы, который проверяет, что поля не пустые, email содержит '@', пароль длиннее 6 символов. Запрос к ИИ на генерацию функции валидации. Вывод сообщений об ошибках на той же странице.
Форма контроля: Контроль результатов творческой деятельности (работающая форма с валидацией).

Тема 59. Сессии и состояние пользователя. Генерация логики входа/регистрации через ИИ
Теория: Понятие сессии (состояние между HTTP-запросами). Flask-сессии (from flask import session). Хранение данных (например, session['user_id']). Реализация входа (логин), выхода (logout). Защита паролей (хэширование).
Практика: Запрос к ИИ: «Реализуй простую систему логина на Flask: хранение пользователей в словаре (временном), хэширование пароля через werkzeug.security, использование сессий». Интеграция кода. Создание страниц логина, дашборда, выхода.
Форма контроля: Контроль результатов творческой деятельности (работающая аутентификация).

Тема 60. Шаблонизатор Jinja2: динамические страницы, циклы, условия. ИИ-генерация шаблонов
Теория: Синтаксис Jinja2: переменные {{ }}, операторы {% %}, условия (if), циклы (for). Наследование шаблонов (extends, block). Фильтры (|length, |upper).
Практика: Запрос к ИИ: «Создай шаблон Jinja2 для отображения списка товаров: заголовок, таблица с колонками (название, цена, наличие), если список пуст – показать сообщение "Товаров нет"». Интеграция шаблона в Flask-приложение, передача списка из маршрута. Создание базового шаблона base.html с меню.
Форма контроля: Контроль результатов творческой деятельности (работающие динамические страницы).

Тема 61. Работа с базой данных (SQLite через Flask-SQLAlchemy). ИИ для создания моделей и запросов
Теория: ORM (SQLAlchemy), модель данных (класс, унаследованный от db.Model). Поля (Integer, String, DateTime). Миграции (Flask-Migrate). Базовые запросы: User.query.filter_by(username=...).first().
Практика: Установка Flask-SQLAlchemy. Запрос к ИИ: «Создай модель User для Flask-SQLAlchemy с полями id, username, email, password_hash, created_at. Напиши код для добавления пользователя в базу и выборки всех пользователей». Реализация. Вывод списка пользователей на отдельной странице.
Форма контроля: Контроль результатов творческой деятельности (база данных создана, данные добавляются и отображаются).

Тема 62. Проектирование веб-игры: идея, игровая механика на сервере (например, текстовый квест или кликер). ИИ для генерации описаний
Теория: Типы веб-игр с серверной логикой: текстовый квест (выбор действий), кликер (счётчик очков), викторина. Проектирование: какие данные хранить на сервере (очки, прогресс, состояние сессии). ИИ для генерации контента (описания локаций, вопросов).
Практика: Команды выбирают тип игры. Запрос к ИИ на генерацию описаний (например, «Сгенерируй 5 локаций для текстового квеста в космосе, укажи описание и возможные действия»). Проектирование структуры базы данных для игры. Создание каркаса (маршруты, шаблоны).
Форма контроля: Наблюдение, утверждение идеи и структуры наставником.

Тема 63. Реализация игровой логики на Python (сервер) + простой интерфейс на HTML/CSS. ИИ как помощник в отладке
Теория: Перенос игровой механики в код: функции для обработки действий игрока, изменения состояния (счёт, здоровье, инвентарь). Отладка с помощью ИИ – отправка ошибок, запрос исправлений.
Практика: Реализация серверной логики выбранной игры (например, для кликера – увеличение счётчика по POST-запросу, для квеста – выбор действия и смена локации). ИИ помогает написать функции. Интерфейс – простой HTML/CSS без JS (обновление страницы при каждом действии). Тестирование, отладка с помощью ИИ.
Форма контроля: Контроль результатов творческой деятельности (работающая базовая версия игры).

Тема 64. Добавление JavaScript на фронтенд: динамические обновления без перезагрузки (fetch API). ИИ для генерации fetch-запросов
Теория: Принципы AJAX. Fetch API: отправка запросов к серверу из JS, обработка ответа (JSON). Обновление DOM без перезагрузки страницы. Сравнение с классическим подходом (перезагрузка).
Практика: : формулировка запроса к AI, предполагающая написание последовательности операций для отправки данных и обновления отображаемого состояния. Встраивание в

интерактивное приложение, где воздействие инициирует передачу сведений, а в ответ происходит модификация внутреннего состояния и возврат формализованного сообщения. Вариативное применение для сценариев с последовательным выбором вариантов и динамическим изменением контекста.
Форма контроля: Контроль результатов творческой деятельности (динамическая игра без перезагрузки).

Тема 65. Стилизация и адаптив: CSS + ИИ. Анимации интерфейса
Теория: Адаптивный дизайн (медиа-запросы, гибкие сетки). CSS-анимации (transition, keyframes). ИИ для генерации стилей и анимаций (промпт: «Создай адаптивный дизайн для игрового интерфейса: кнопки, панель счёта, карточки локаций»).
Практика: Запрос к ИИ на генерацию CSS для улучшения внешнего вида игры. Добавление медиа-запросов для мобильных устройств. Реализация анимации наведения кнопок, появления очков. Интеграция в проект.
Форма контроля: Контроль результатов творческой деятельности (стилизованная адаптивная игра).

Тема 66. Подготовка к публикации: переменные окружения, requirements.txt, статические файлы. ИИ для генерации инструкций
Теория: Переменные окружения (секретный ключ, настройки БД). Файл .env, библиотека python-dotenv. Создание requirements.txt (заморозка зависимостей). Структура статических файлов (CSS, JS, изображения).
Практика: Запрос к ИИ: «Создай инструкцию по подготовке Flask-приложения к деплою: список переменных окружения, команды для создания requirements.txt, настройка статики». Выполнение инструкции: создание .env, генерация requirements.txt (pip freeze), проверка, что статические файлы подгружаются правильно.
Форма контроля: Контроль результатов творческой деятельности (файл requirements.txt, корректная работа приложения с переменными окружения).

Тема 67. Публикация на PythonAnywhere (или аналогичном хостинге). Деплой веб-игры. ИИ для решения проблем при публикации
Теория: Обзор бесплатных хостингов для Flask (PythonAnywhere, Render, Railway). Этапы деплоя: загрузка кода, настройка виртуального окружения, указание WSGI-файла, настройка переменных окружения. Возможные проблемы (пути к файлам, версии Python).
Практика: Регистрация на PythonAnywhere. Загрузка проекта через Git или вручную. Настройка веб-приложения. Запуск игры в интернете. При возникновении ошибок – запрос к ИИ (текст ошибки), получение рекомендаций.
Форма контроля: Контроль результатов творческой деятельности (игра доступна по публичной ссылке).

Тема 68. Тестирование опубликованной игры. Сбор обратной связи. Итеративные улучшения с помощью ИИ
Теория: Методы тестирования веб-приложения (функциональное, кроссбраузерное, нагрузочное). Сбор обратной связи от одногруппников (опрос, комментарии). Приоритизация улучшений. ИИ для генерации идей по улучшению.
Практика: Учащиеся играют в проекты друг друга, выявляют баги и дают предложения. Каждая команда собирает обратную связь (например, в Trello или Google Forms). Запрос к ИИ: «Предложи 3 улучшения для моей веб-игры [описание]». Реализация одного-двух улучшений (итерация).
Форма контроля: Наблюдение, отчёт о внесённых улучшениях.

Тема 69. Защита проекта: демонстрация работающей онлайн-игры, презентация использования ИИ на всех этапах. Рефлексия

Теория: Структура защиты: URL игры, описание механик, демонстрация (2–3 минуты геймплея), рассказ о том, как использовался ИИ (генерация кода, отладка, контент, стили), рефлексия (что далось легко, что трудно).
Практика: Публичная защита проектов (по 7–10 минут на команду). Демонстрация живой онлайн-игры. Ответы на вопросы. Заполнение рефлексивного листа.
Форма контроля: Защита проекта (оценка наставником и взаимная оценка).

Тема 70. Обобщение тем курса. Обсуждение вектора дальнейшего развития
Теория: Систематизация знаний, полученных во втором году (командная разработка, событийное программирование в «Берлоге», 3D-моделирование, Godot + ИИ, веб-разработка с ИИ). Связь между модулями. Возможные пути продолжения: участие в хакатонах, углублённое изучение движков, фулстек-разработка, DevOps.
Практика: Групповая дискуссия «Какой модуль был самым полезным и почему?». Составление личного плана на следующие 6 месяцев (курсы, проекты, чтение).
Форма контроля: Беседа, наблюдение.

Тема 71. Подготовка к итоговой выставке
Теория: Требования к выставочному стенду: ноутбук/планшет с демо, плакат (описание проекта), QR-код на игру (для веб-проектов). Краткая речь (1–2 минуты). Рекомендации по визуальному оформлению.
Практика: Команды готовят выставочные материалы: создают постер (Canva или PowerPoint), продумывают интерактивную демонстрацию, репетируют презентацию. Проведение внутреннего прогона.
Форма контроля: Контроль результатов творческой деятельности (готовые стендовые материалы).

Тема 72. Проведение итоговой выставки проектов
Теория: Организация выставки (зоны, тайминг, жюри). Критерии оценки: техническая сложность, дизайн, работа ИИ, презентационные навыки. Правила голосования.
Практика: Участвуют все модульные проекты (каждый выбирает один или несколько). Презентация посетителям (педагогам, родителям, другим учащимся). Ответы на вопросы. Подведение итогов, награждение.
Форма контроля: Защита проекта (оценка внешним жюри, приз зрительских симпатий).

По окончанию 2 года обучения учащиеся будут знать и уметь:

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ
Личностные результаты:

- Обучающийся умеет использовать компьютер и цифровые инструменты (включая ИИ) для решения прикладных проектных задач, поиска и критического анализа информации в сети Интернет.

- Обучающийся выполняет проектные и исследовательские работы в области разработки визуальных новелл, веб-приложений, алгоритмических задач и программных продуктов с использованием искусственного интеллекта.
- У обучающихся развиты личностные качества в процессе коллективной деятельности над проектом: ответственность, инициативность, дисциплинированность, способность доводить начатое до конца.
- Сформирована потребность в сотрудничестве со сверстниками, доброжелательное отношение к членам команды, бесконфликтное поведение, умение прислушиваться к мнению товарищей по команде и аргументированно отстаивать свою позицию.
- Осуществлено духовно-нравственное, гражданско-патриотическое, военно-патриотическое и трудовое воспитание через тематику проектных заданий (в том числе создание новелл на историческую или социально значимую тему) и осознание ценности интеллектуального труда.
- Сформировано уважительное отношение к интеллектуальной собственности, понимание лицензий (Creative Commons, Open Source) и правил использования готовых графических, музыкальных и программных материалов, а также результатов работы ИИ.

Предметные результаты:
Обучающийся научился:
- **Работать на компьютере** в различных текстовых редакторах (VS Code, PyCharm, блокнот) и интегрированных средах разработки, настраивать их под свои задачи.
- **Устанавливать и настраивать браузер** в режиме разработчика (инструменты разработчика Chrome/Firefox) для отладки HTML/CSS/JS, анализа сетевых запросов и проверки адаптивности интерфейса.
- **Работать в сети Интернет** эффективно: осуществлять поиск информации по ключевым словам, фильтровать результаты, проверять источники на достоверность, сохранять и систематизировать найденные материалы (изображения, звуки, библиотеки кода, документацию).
- **Разрабатывать и отлаживать веб-страницы и программы** на языках **HTML**, **CSS**, **JavaScript** (в том числе с использованием движка визуальных новелл Monogatari): создавать структуру страницы, стилизовать интерфейс, прописывать диалоги и выборы, использовать переменные, условия, события и анимации.
- **Создавать и обрабатывать растровые и векторные изображения** (спрайты персонажей, фоны) в графических редакторах (GIMP, Krita, Inkscape) и подготавливать их для интеграции в игру.
- **Писать программы на языке Python** (переменные, типы данных, условные операторы, циклы, списки, словари, функции) и применять их для решения вычислительных и алгоритмических задач.
- **Оценивать сложность алгоритмов** (O-нотация), реализовывать базовые алгоритмы поиска (линейный, бинарный), сортировки (пузырьком, вставками, слиянием), рекурсивные функции, обход графов (DFS, BFS), простейшие алгоритмы динамического программирования.
- **Формулировать техническое задание**, разбивать проект на итерации, использовать минимально работоспособный продукт (MVP) и прототипирование, документировать результаты (руководство пользователя, техническое описание).
- **Использовать инструменты искусственного интеллекта** (чат-боты, генераторы кода, графические нейросети) как вспомогательные средства при проектировании, написании кода, генерации идей, составлении документации и тестировании, при этом критически оценивая результаты их работы.
- **Выполнять проектные и исследовательские работы** в команде: от генерации идеи и постановки цели до публичной защиты продукта, включая самоанализ (рефлексию) и планирование дальнейшего развития.

Метапредметные результаты:
Обучающийся умеет:

- **Ставить цель и задачи** для реализации проектов на основе анализа потребностей и имеющихся ресурсов, используя методы SMART.
- **Преобразовывать практическую задачу** (создание игры, приложения, алгоритма) в познавательную (как это работает, какие технологии выбрать) — самостоятельно или с помощью педагога.
- **Анализировать задачи** по разработке программного продукта, разбивая их на известные подзадачи (отрисовка интерфейса, написание скрипта, отладка) и выделяя новое, требующее дополнительного изучения.
- **Синтезировать из известного материала новую информацию** (комбинировать команды CSS, функции JavaScript, библиотеки Python), оценивая её актуальность и востребованность для своего проекта.
- **Планировать собственную деятельность** (составлять дорожную карту проекта, график итераций, список задач на спринт) в соответствии с поставленной задачей и условиями её реализации, действовать в соответствии с планом.
- **Контролировать и оценивать свои действия** (проводить модульное и интеграционное тестирование, анализировать ошибки, измерять время выполнения алгоритмов) и вносить коррективы в их выполнение.
- **Пользоваться компьютерными источниками информации** (официальная документация, форумы, базы знаний, онлайн-курсы, генеративные ИИ-модели) для решения возникающих технических проблем и расширения кругозора.
- **Организовывать своё рабочее (учебное) место** (удобное расположение окон редактора, браузера, консоли, системы контроля версий) и поддерживать порядок в файловой структуре проекта.
- **Соблюдать правила безопасности** при работе за компьютером, в сети Интернет (защита персональных данных, проверка ссылок, использование лицензионного ПО) и при использовании ИИ-сервисов.

Познавательные Обучающийся умеет:

- **Подбирать и анализировать специальную литературу** (документацию по HTML/CSS/JS, Python, Monogatari, алгоритмам), выделяя главное и применимое к своей задаче.
- **Осуществлять учебно-исследовательскую работу** (сравнение алгоритмов сортировки по времени работы, анализ эффективности разных подходов к хранению игровых данных, тестирование влияния размера изображений на производительность новеллы).
- **Понимать информацию, представленную в разных формах**: текст (техническое задание, баги-репорты), рисунки (схемы ветвлений сюжета, диаграммы классов, скринкасты), схемы (алгоритмические блок-схемы, архитектура проекта, дерево переходов), таблицы (Big O сложностей, результаты тестирования).
- **Оценивать актуальность и достоверность информации**, представленной в сети Интернет: отличать устаревшую документацию от современной, проверять репутацию источников, сверять данные с официальными сайтами и несколькими независимыми ресурсами, распознавать сгенерированный ИИ текст, требующий проверки.
- **Умеет работать в команде, распределяя роли** (сценарист, художник, программист, тестировщик), согласовывать интерфейсы между компонентами, проводить ретроспективы и давать обратную связь товарищам.
- **Способен публично представить результаты проекта** (защита прототипа, продуктового результата), аргументированно ответить на вопросы аудитории и комиссии.

РАЗДЕЛ 2. КОМПЛЕКС ОРГАНИЗАЦИОННО-ПЕДАГОГИЧЕСКИХ УСЛОВИЙ

Календарный учебный график

Год обучения	Дата начала обучения по программе	Дата окончания обучения по программе	Всего учебных недель	Количество учебных часов	Режим занятий
1	1 сентября	31 мая	36	144	2 раза в неделю по 2 часа
2	1 сентября	31 мая	36	144	2 раза в неделю по 2 часа
Продолжительность каникул		С 31 декабря по 10 января текущего года			
		С 1 июня по 31 августа текущего года			

УСЛОВИЯ РЕАЛИЗАЦИИ ПРОГРАММЫ

Требования к помещению для занятий

В соответствии с Санитарно-эпидемиологическими правилами и нормативами СанПиН 2.4.3648-20 для организации учебного процесса имеется кабинет из расчета 2 квадратных метра на каждого учащегося, с возможностью проветривания и зонирования пространства для групповой работы.

Требования к мебели:

1) стандартные, комплектные и с маркировкой, соответствующей ростовой группе, учебные столы и стулья, согласно требованиям СанПиН 2.4.3648-20;

2) стеллаж, стенд для выставки книг и иных материалов.

Кадровое обеспечение программы

Программа реализуется педагогом дополнительного образования, имеющим профессиональное образование в области, соответствующей профилю программы, и постоянно повышающим уровень профессионального мастерства.

Материально-техническое обеспечение:

С целью создания оптимальных условий для формирования интереса у детей к конструированию с элементами программирования, развития конструкторского мышления, была создана предметно-развивающая среда:

- столы, стулья (по росту и количеству детей);
- интерактивная доска;
- технические средства обучения (ТСО) (мультимедийное устройство).

Формы организации учебного занятия

Содержание программы включает в себя занятия разных типов, на которых решаются вокальные, творческие и воспитательные задачи. Форма проведения занятия варьируется, в рамках одного занятия сочетаются разные ***виды деятельности:***

- индивидуальная;
- групповая;
- работа в парах;
- фронтальная;
- индивидуально-групповая;
- работа по подгруппам.

На теоретических занятиях применяются методы, способствующие первичному усвоению учебного материала:

- систематизация знаний;
- глубокое изучение предмета;
- пошаговое освоение учебного материала;
- использование материала всех предыдущих разделов.

На практических занятиях применяются методы, способствующие закреплению и совершенствованию приобретенных знаний: упражнения, практические занятия. Степень

самостоятельности при выполнении практических занятий постепенно повышается. При проведении занятий так же используются демонстрационные и обучающие программы, раздаточный (дидактический) материал.

Используются следующие ***формы*** занятий:

1. *По количеству детей:* групповые, коллективные.

2. *По особенностям коммуникативного взаимодействия педагога и детей:* мастер-класс, соревнование, викторина, «мозговой штурм», открытое занятие, защита проектов.

3. *По дидактической цели:* вводное занятие; занятие по углублению знаний; практическое занятие; занятие по контролю знаний, умений и навыков; комбинированные формы занятий.

Типы занятий:

Основными типами занятий по программе «IT- КВАНТУМ. В компьютерною разработку» являются:

- теоретический;
- практический;
- контрольный.

Педагогическая деятельность в группах проводится с учётом возрастных особенностей детей.

Учебно-методическое и информационное обеспечение:

- инструктажи по охране труда и технике безопасности;
- учебная и методическая литература;
- методические пособия и разработки;
- разноуровневые задания и упражнения;
- сценарии воспитательных мероприятий;
- сборник подвижных игр;
- тематические презентации;
- образовательные электронные ресурсы;
- Интернет-ресурсы.

ФОРМЫ АТТЕСТАЦИИ

Формы, порядок и периодичность аттестации обучающихся определяются ГБОУ «ДАТ «Солнечный город» самостоятельно.

Виды контроля: *входной, текущий, промежуточный, итоговый.*

Входной контроль (проверка знаний учащихся на начальном этапе освоения Программы). Проводится в начале реализации Программы *в форме* опроса, педагогического наблюдения. Отдельно проводится тестирования с целью выявления склонности учащегося и установок на учебу.

Текущий контроль (отслеживание активности обучающихся на занятии). Текущим контролем является диагностика, проводимая по окончанию каждого занятия результатов творческой деятельности и степени самостоятельности выполнения работ. Кроме того отслеживается дополнительная работа учащихся.

Промежуточный контроль (подведение промежуточных итогов). Проводится по окончанию каждого модуля в виде презентации творческих работ учащихся.

Итоговый контроль (заключительная проверка знаний, умений, навыков по итогам реализации Программы в каждом учебном году). Итоговый контроль по темам проходит в виде проектных заданий, творческого конструирования, защиты презентаций. Результаты контроля фиксируются в протоколах.

Средства контроля

Контроль знаний, умений и навыков обучающихся обеспечивает оперативное управление учебным процессом, и выполнят обучающую, проверочную, воспитательную и

корректирующую функции. Показателем эффективности любого процесса служит конечный результат.

Формы контроля:

- опрос;
- наблюдение;
- коллективная работа;
- практические упражнения;
- творческие задания;
- защита проекта.

Сроки проведения:

- сентябрь – входящая диагностика и контроль;
- декабрь - текущая диагностика и контроль;
- апрель-май - итоговая диагностика и контроль.

Результативность обучения дифференцируется по уровням: высокий, средний, низкий.

Критерии оценок

Уровень освоения образовательной программы оценивается по следующим показателям:

1. Имеет представление об основных командах и операторах Python
2. Умеет формализировать текстовую задачу и перевести её в код
3. Работает в команде.
4. Создает программы по готовым требованиям.
5. Может рассказать о своем замысле, описать ожидаемый результат, назвать способы конструирования модели, продемонстрировать ее технические возможности.

Определены следующие уровни сформированности умения обучающихся писать программы

Низкий уровень (0 – 1 баллов). Обучающийся воспроизводит готовый программный код с ошибками. Не умеет их самостоятельно устранять и устраняет только после подсказки учителя или под его прямым руководством.

Удовлетворительный уровень (2 – 3 баллов). Обучающийся безошибочно воспроизводит готовый программный код. Умеет пользоваться отладчиком и может исправлять ошибки путём сопоставления с готовым исходным кодом.

Средний уровень (7 – 9 баллов). Умеет самостоятельно писать программный код, пользуясь готовыми алгоритмами или модифицируя готовый код под свои нужды. Пользуется отладчиком уверенно. Может выявлять не только синтаксические, но и алгоритмические ошибки.

Высокий уровень (10 – 12 баллов). Обучающийся умеет самостоятельно писать код по готовому техзаданию или переводить текстовые задачи в алгоритмическую форму. Может тестировать код в критических местах.

Отличный уровень (13 – 15 баллов). Обучающийся проявляет самостоятельный интерес к написанию программного кода. Может самостоятельно изыскивать задачи, их формализовать и находить оптимальные решения. Имеет стратегию тестирования кода и придерживается ей.

МЕТОДИЧЕСКОЕ И ДИДАКТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ

Методы обучения:

- словесные (лекция, рассказ, беседа);
- наглядные (тематические презентации);
- практические (упражнения, игры, задания, выполнение индивидуальных и групповых заданий, занятия с элементами тренинга).

Методы организации учебно-воспитательного процесса и используемые технологии

Для реализации задач и содержания программы используется ряд основных **методов и приёмов:**

- **информационно-познавательные** – беседы, просмотр видеофильмов;
- **практические** – демонстрация способов действий педагогом, воспроизведение действий обучающимися;
- **творческие** – конструирование, импровизация, проведение тематических выставок;
- **индивидуальные, групповые, коллективные** приемы работы;
- **познавательный** (восприятие, осмысление и запоминание нового материала с привлечением наблюдения готовых примеров, моделирования, изучения иллюстраций, восприятия, анализа и обобщения демонстрируемых материалов);
- **метод проектов** (при усвоении и творческом применении навыков и умений в процессе разработки собственных моделей);
- **систематизирующий** (беседа по теме, составление схем и т.д.);
- **контрольный метод** (при выявлении качества усвоения знаний, навыков и умений и их коррекция в процессе выполнения практических заданий);
- **групповая работа** (используется при совместной разработки программ, а также при разработке проектов);
- **соревнования** (практическое участие детей в разнообразных мероприятиях по техническому конструированию).

Методы воспитания:

- убеждение - это метод воспитания, который выражается в эмоциональном и глубоком разъяснении сущности социальных и духовных отношении, норм и правил поведения;
- поощрение – это метод воспитания, стимулирующий деятельность учащегося. Поощрение вызывает положительные эмоции, способствовавшее возникновению чувства уверенности ребенка в своих силах;
- упражнение - это метод воспитания, который предполагает такую организацию деятельности, которая позволяет учащимся накапливать привычки и опыт правильного поведения, связывать слово с делом, убеждение с поведением.
- контроль - это метод воспитания, заключается в наблюдении за деятельностью и поведением учащихся с целью побуждения их к соблюдению установленных правил, а также к выполнению определенных заданий.

Педагогические технологии:

В работе используются различные **педагогические технологии**:

- индивидуальное обучение;
- личностно-ориентированный подход;
- дифференцированное обучение
- развивающее обучение;
- здоровьесберегающие;
- игровые технологии;
- информационно-коммуникационные

Основной педагогической технологией в системе Кванториумов является метод проектов.

Технология проектного обучения, соответствующая всем современным нормам, которые предъявляются к учебному процессу, в полной мере способствует решению поставленных задач и характеризуется:

- личностной ориентацией — развитие личностных способностей детей, индивидуализация их обучения, при которой учитываются интересы, умения и желания ребят;
- приумножением деятельностной составляющей — формирование навыков действовать (поиск и обработка информации, проведение анализа, оценка, принятие решения);

- сочетанием теоретических и практических занятий;
- творческой деятельностью (создание конечного продукта — собственного творения).

Актуальность методики проектного обучения заключается в многофункциональной направленности и возможности процесса её интеграции в целостную образовательную систему, в процессе которой, наравне с освоением предметных знаний, учащиеся разносторонне развиваются.

При реализации любой, в том числе базовой, образовательной программы проектная деятельность нацелена на развитие именно инженерного мышления, которое охватывает не только информацию, факты, материалы и формулы, но и основывается на способности самостоятельно выстроить алгоритм действий, логическую цепочку создания программного продукта, т.е. на выполнении следующих основных инженерных заданий:

- поиск действенного способа достижения ожидаемого результата:
- процесс проектирования:
- расчёт показателей;
- программное воплощение;
- испытания и доработка;
- представление готового программного продукта.

Благодаря такой деятельности обучающиеся получают знания и навыки в процессе самостоятельного проектирования и выполнения постепенно усложняющихся практических заданий.

Здоровьесберегающие технологии:

В ГБОУ «ДАТ «Солнечный город» Министерства просвещения и науки КБР уделяется большое внимание комфортному пребыванию учащихся в учебном заведении, учебный процесс построен с использованием здоровьесберегающих технологий. Внедряемое в ЦДОД здоровьесберегающее образование можно рассматривать как процесс воспитания и обучения, результатом которого является достижение учащимися уровня образованности без ущерба своему здоровью. В дополнительном образовании в учебном процессе используется перспективный путь – применение полученных знаний в любимом деле для самореализации личности ребёнка. Следовательно, в дополнительном образовании снимаются проблемы, связанные с необходимостью усваивать большое количество информации в ограниченное время. Что само по себе благоприятно сказывается на состоянии здоровья. Занятия в ЦДОД рассчитаны так, чтобы учащийся не испытывал нагрузки, а в процессе творчества развивался без ущерба для здоровья. Здоровый и духовно развитый ребёнок счастлив – он отлично себя чувствует, получает удовлетворение от своей работы, стремится к самоусовершенствованию, развивая себя всесторонне в дополнительном образовании.

СПИСОК ЛИТЕРАТУРЫ

Литература для педагога:

1. Бизли Д. Python. Книга рецептов. – М.: ДМК Пресс, 2021. – 464 с.
2. Браун М. Python для детей. Руководство по программированию. – М.: Манн, Иванов и Фербер, 2021. – 320 с.
3. Бриггс Д. Python для детей. Самоучитель по программированию. – М.: Манн, Иванов и Фербер, 2022.
4. Гаррисон К. Программируем с Python. Библиотека Turtle. – М.: ДМК Пресс, 2022. – 200 с.
5. Дронов В. Python 3 и PyQt 5. Разработка приложений. – М.: БХВ, 2021.
6. Лутц М. Изучаем Python. – 5-е изд. – СПб.: Символ-Плюс, 2022. – 1648 с.
7. Матиясевич М. Программирование на Python в примерах и задачах. – М.: Эксмо, 2022. – 320 с.
8. Мэтиз Э. Изучаем Python. Программирование игр, визуализация данных, веб-приложения. – СПб.: Питер, 2023.
9. Сафронов Г. Python и Pygame. Разработка игр. – СПб.: БХВ, 2021. – 400 с.
10. Свейгарт Э. Python. Чистый код для продолжающих. – СПб.: Питер, 2023. – 352 с.
11. Теллез А. Python в инфографике. – М.: Эксмо, 2023. – 180 с.
12. Щерба А. В. Программирование на Python: первые шаги. – 2-е изд. – М.: Лаборатория знаний, 2024. – 250 с.
13. Занимательные задачи по программированию на Python. – М.: Лаборатория знаний, 2022.
14. Пашковская Ю. В. Программирование на Scratch для детей. Уровень 1. – М.: Лаборатория знаний, 2024. – 224 с.
15. Разработка web-страниц на HTML, CSS и JavaScript: учебное пособие для вузов. – 2024. – 148 с.
16. Харбанс Р. Грокаем алгоритмы искусственного интеллекта. – СПб.: Питер, 2024. – 366 с.
17. Фаррелл Д. Python. Как стать профессионалом. Разработка приложений с Python и Flask. – М.: БХВ, 2024.

Литература для обучающихся

1. Джейсон Бриггс Python для детей. Самоучитель по программированию. – М.: Манн, Иванов и Фербер, 2022.
2. Эрик Мэтиз Изучаем Python. Программирование игр, визуализация данных, веб-приложения. – СПб.: Питер, 2023)
3. Владимир Дронов Python 3 и PyQt 5. Разработка приложений. – М.: БХВ, 2021.
4. Коллектив авторов Занимательные задачи по программированию на Python. – М.: Лаборатория знаний, 2022. (Олимпиадные задачи для углубленного изучения)

Онлайн-курсы и интерактивные платформы

Code.org – https://code.org/ (Базовые уроки по Python для детей)
Trinket (Turtle онлайн) – https://trinket.io/python (Можно писать и запускать код прямо в браузере)
CheckiO – https://checkio.org/ (Игровое обучение Python)
Учебник на Python.org – https://docs.python.org/3/tutorial/ (Официальная документация)

Бесплатные книги и учебники

"Укус Питона" (для новичков) – https://wombat.org.ua/AByteOfPython/
"Python для всех" (на русском) – https://www.py4e.com/book

Приложение №1
к дополнительной
общеразвивающей программе
«IT-КВАНТУМ. ВВЕДЕНИЕ В
КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ И НАУКИ
КАБАРДИНО-БАЛКАРСКОЙ РЕСПУБЛИКИ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«ДЕТСКАЯ АКАДЕМИЯ ТВОРЧЕСТВА «СОЛНЕЧНЫЙ ГОРОД»

РАБОЧАЯ ПРОГРАММА НА 2026-2027 УЧЕБНЫЙ ГОД К ДОПОЛНИТЕЛЬНОЙ ОБЩЕРАЗВИВАЮЩЕЙ ПРОГРАММЕ «IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»

Уровень программы: базовый
Адресат: 11-15 лет
Год обучения: первый, второй
Автор-составитель: Кравченко Алексей Михайлович,
педагог дополнительного образования

Нальчик, 2026

Цели программы: ранняя профессиональная ориентация обучающихся, повышение престижа инженерных и научных профессий, подготовка кадрового резерва для глобального технологического лидерства России в области информационных технологий.

Задачи программы:

Личностные задачи:

- сформировать и развить интерес к инженерным специальностям;
- научить обучающихся выполнять проектные и исследовательские работы;
- развить личностные качества в процессе коллективной деятельности над проектом;
- способствовать социализации и адаптации обучающихся детей к жизни в обществе;
- обеспечить духовно-нравственное, гражданско-патриотическое, военно-патриотическое и трудовое воспитание детей и подростков.

Предметные задачи

обучающийся должен научиться:

- работать на компьютере в различных текстовых редакторах и IDE;
- устанавливать и настраивать браузер в режиме разработчика;
- работать в сети Интернет (поиск, анализ, сохранение информации);
- разрабатывать и отлаживать программы на языке Python
- выполнять проектные исследовательские работы;

Метапредметные задачи:

Регулятивные:

научить обучающихся:

- ставить цель и задачи для реализации проектов;
- преобразовывать практическую задачу в познавательную самостоятельно или с помощью педагога;
- анализировать задачи, разбивая их на известные подзадачи и вычленяя новое;
- синтезировать из известного материала новую информацию, оценивая ее актуальность и востребованность;
- планировать собственную деятельность в соответствии с поставленной задачей и условиями ее реализации, действовать в соответствии с планом;
- контролировать и оценивать свои действия и вносить коррективы в их выполнение;
- пользоваться компьютерными источниками информации;
- организовывать свое рабочее (учебное) место;
- обучить навыкам соблюдения правил безопасности в процессе деятельности.

Познавательные:

научить обучающихся:

- подбирать и анализировать специальную литературу;
- осуществлять учебно-исследовательскую работу;
- понимать информацию, представленную в виде текста, рисунков, схем;
- оценивать актуальность и достоверность информации представленной в Сети Интернет.

ПЛАНИРУЕМЫЕ РЕЗУЛЬТАТЫ

Личностные результаты:

- обучающийся умеет использовать компьютер в повседневной работе (искать и анализировать информацию в Сети Интернет);
- обучающийся выполняет проектные и исследовательские работы;
- у обучающихся развиты личностные качества в процессе коллективной деятельности над проектом;

- сформировано духовно-нравственное, гражданско-патриотическое, военно-патриотическое и трудовое воспитание детей и подростков

Предметные результаты:

обучающийся научился:

- работать на компьютере в различных текстовых редакторах и IDE;
- устанавливать и настраивать браузер в режиме разработчика;
- работать в сети Интернет (поиск, анализ, сохранение информации);
- разрабатывать и отлаживать программы на языке Python
- выполнять проектные исследовательские работы;

Метапредметные результаты:

Регулятивные:

обучающийся умеет:

- ставить цель и задачи для реализации проектов;
- преобразовывать практическую задачу в познавательную самостоятельно или с помощью педагога;
- анализировать задачи, разбивая их на известные подзадачи и вычленяя новое;
- синтезировать из известного материала новую информацию, оценивая ее актуальность и востребованность;
- планировать собственную деятельность в соответствии с поставленной задачей и условиями ее реализации, действовать в соответствии с планом;
- контролировать и оценивать свои действия и вносить коррективы в их выполнение;
- пользоваться компьютерными источниками информации;
- организовывать свое рабочее (учебное) место;
- обучить навыкам соблюдения правил безопасности в процессе деятельности.

Познавательные:

обучающиеся умеют:

- подбирать и анализировать специальную литературу;
- осуществлять учебно-исследовательскую работу;
- понимать информацию, представленную в виде текста, рисунков, схем;
- оценивать актуальность и достоверность информации представленной в Сети Интернет.

Календарно-тематический план
к дополнительной общеразвивающей программе
«IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»
1 год обучения

№	Дата проведения занятия		Название раздела, темы	Количество часов	Содержание деятельности		Форма аттестации
	по плану	по факту			Теоретическая часть занятия	Практическая часть занятия	
Первый год Модуль 1: Введение в визуальные новеллы							
1			**Техника безопасности. Проектная деятельность в системе «Кванториумов»**	2	Техника безопасности при работе с компьютерами. Кванториумы. Цели и задачи Кванториумов. Национальная технологическая инициатива. Роль IT-квантума. Чем работа в кванториуме отличается от обучения в школе или занятий в кружке. Проектная деятельность. Отличительные особенности проектной работы. Потоки создания ценности: продуктовый и образовательный. Виды проектов. Кейсы. Команда и её роль в проекте.	Дискуссия на заданную тему. Выбор проекта для реализации. Формулировка продуктового результата проекта. Опрос – "каких образовательных результатов планируют достичь учащиеся в ходе проектной деятельности". Тренинг командообразования.	беседа, наблюдение
2			**Что такое игра. Жанры видеоигр**	2	Определение понятия «игра». Игра как система правил, вызовов и наград. Классификация видеоигр по жанрам: экшен, стратегия, RPG, симуляторы, головоломки, визуальные новеллы и др. Гибридные жанры. Особенности	Интерактивная викторина «Угадай жанр по скриншоту и описанию». Работа в малых группах: выбрать 3 разных жанра и привести по 2 примера игр, объяснить, почему они относятся к этим жанрам.	устный опрос, выполнение группового задания.

					каждого жанра: механики, целевая аудитория, примеры.		
3			**Новеллы. Концепция новеллы: идея, тема, логлайн**	2	Что такое визуальная новелла: отличия от книги, фильма и классической RPG. Ключевые элементы: текст, графика, выборы, ветвления. Понятия идеи (что я хочу рассказать), темы (о чём эта история глубже), логлайна (одно-два предложения, передающих суть). Примеры удачных логлайнов известных новелл.	Мозговой штурм — каждый придумывает 3 идеи для будущей новеллы. Формулирует к одной из них тему и логлайн. Презентация своего логлайна группе, обратная связь.	контроль результатов творческой деятельности
4			**Игровая команда. Разработка игры**	2	Роли в игровой команде: продюсер, геймдизайнер, сценарист, художник, программист, тестировщик, звукорежиссёр. Распределение обязанностей. Особенности инди-разработки (соло или малая команда, совмещение ролей). Этапы разработки игры: пре-продакшн, продакшн, тестирование, релиз, пост-релизная поддержка.	Проектная сессия — каждая команда (3-4 человека) распределяет роли для своего проекта. Составление дорожной карты проекта на ближайшие 4 недели. Инструменты для планирования (Trello, Notion, Miro).	контроль результатов творческой деятельности
5			**Игровой баланс**	2	Что такое баланс в игре. Почему важно, чтобы игра была честной, но интересной. Виды баланса: сложность, темп, ресурсы, экономика. Понятия «кривая сложности», «онбординг», «туториал». Баланс в визуальных новеллах: влияние выборов, доступность всех веток, избегание «тупиков» и «мёртвых концов».	Анализ известной новеллы (например, «Бесконечное лето» или Doki Doki Literature Club) на предмет баланса: где сложность растёт плавно, где скачкообразно. Проектирование простой системы баланса для своего прототипа (шкала отношений героев, очки действий).	контроль результатов творческой деятельности

6			**Интерактивный сторителлинг**	2	Определение интерактивного сторителлинга. Отличие от линейного повествования. Способы вовлечения игрока: выбор, последствия, иллюзия свободы, моральные дилеммы. Принцип «agency» (способность игрока влиять на мир). Понятие «водяной знак» сюжета — общая нить, несмотря на ветвления.	Разбор конкретных сцен из новелл, где выбор игрока действительно меняет ход событий. Создание небольшой диалоговой сцены (5-7 реплик) с двумя вариантами ответа, каждый из которых ведёт к разной реакции персонажа.	контроль результатов творческой деятельности
7			**Сеттинг и сюжет**	2	Сеттинг: время, место, законы мира (магия, технологии, социальный строй). Сюжет: последовательность событий. Как сеттинг влияет на сюжет и наоборот. Типы сеттингов: фэнтези, научная фантастика, постапокалипсис, современность, исторический и т.д. Конфликт как двигатель сюжета: внутренний и внешний.	Описание сеттинга для своего проекта (2-3 абзаца: где и когда, какие правила, кто живёт). Построение простой сюжетной арки из 5 точек (завязка, инцидент, развитие, кульминация, развязка). Взаимная проверка в парах.	контроль результатов творческой деятельности
8			**Структура повествования: линейные и нелинейные подходы**	2	Линейное повествование (игрок не влияет на порядок событий). Нелинейное: ветвящийся сюжет, множественные концовки, открытый мир, хабовая система (общие узлы с разными выходами). Преимущества и недостатки каждого подхода. Сложность реализации нелинейных структур для автора и тестировщика. Понятие «дерево диалогов».	Построение схемы сюжета для своего прототипа: линейный, с одной развилкой или с несколькими концовками. Использование диаграммы в Miro или на бумаге. Анализ чужой схемы (работа в парах).	контроль результатов творческой деятельности
9			**Библия персонажей**	2	Что такое «библия персонажа»: документ, описывающий внешность,	Создание библии для 2-3 ключевых персонажей своего	контроль результатов

					характер, привычки, мотивацию, бэкстори, речевые особенности. Зачем нужна библия: для единообразия написания диалогов, для художников, для избегания противоречий. Примеры структур библии.	проекта (имя, возраст, цель, слабости, фраза-характеристика, внешность). Презентация одного персонажа группе.	творческой деятельности
10			**Конфликт в структуре сюжета**	2	Природа конфликта: без конфликта нет истории. Типы конфликтов: человек против человека, человека против природы, человека против общества, человека против себя (внутренний). Конфликт как источник выборов в новелле. Эскалация конфликта. Разрешение конфликта.	Определить главный конфликт своего проекта. Написать короткую сцену (5-10 реплик), где конфликт проявляется и требует от игрока решения (два варианта выбора). Обсуждение в группе: насколько выборы отражают суть конфликта.	контроль результатов творческой деятельности
11			**Решения и последствия**	2	Механика «причина-следствие» в визуальных новеллах. Скрытые и явные последствия. Краткосрочные (немедленная реакция) и долгосрочные (влияют на финал) последствия. Как проектировать последствия, чтобы они не были слишком предсказуемыми или, наоборот, случайными. Понятие «вес выбора».	К каждой из двух развилок своего сюжета прописать минимум одно краткосрочное и одно долгосрочное последствие. Заполнить таблицу «Выбор → Немедленный результат → Отсроченный результат».	контроль результатов творческой деятельности
12			**Типы игровых интерфейсов**	2	UI (пользовательский интерфейс) в визуальных новеллах: окно диалога, кнопки выбора, меню сохранения, экран настроек, инвентарь (если есть), журнал сообщений. Типы интерфейсов: минималистичный,	Анализ интерфейсов 2-3 известных новелл (назвать, что удачно, что нет). Создание эскиза UI для своего проекта на бумаге или в Figma: расположение текстового окна,	контроль результатов творческой деятельности

					кастомный (стилизованный под сеттинг), с анимацией. Принципы usability: понятность, отзывчивость, отсутствие перегруза.	кнопок, портретов персонажей.	
13			**Прототипирование UI**	2	От эскиза к интерактивному прототипу. Инструменты: Figma, Adobe XD, даже PowerPoint. Создание кликабельного прототипа для проверки логики переходов. Важность прототипирования перед программированием.	В Figma создать кликабельный прототип главного экрана новеллы (окно диалога, 2 кнопки выбора, меню). Протестировать на соседе: понятно ли, куда нажимать? Внести правки.	контроль результатов творческой деятельности
14			**Реализация новеллы**	2	Обзор способов создания визуальной новеллы: готовые движки (Ren'Py, Monogatari, TyranoBuilder), программирование на Unity/Godot, конструкторы (Visual Novel Maker). Критерии выбора: сложность, возможности, необходимость программирования. Структура типового проекта: папки со спрайтами, фонами, музыкой; файлы скриптов.	Установка выбранного движка (на примере Monogatari). Запуск демо-проекта. Обзор папок и файлов проекта. Создание первой тестовой сцены с двумя репликами.	контроль результатов творческой деятельности
15			**Возможности конструктора**	2	Базовые команды движка: показать текст, показать фон, показать спрайт, воспроизвести музыку, создать выбор. Хранение переменных (очки отношений). Работа с сохранениями. Плюсы и минусы конструктора.	В своём проекте реализовать сцену из 5-7 реплик с двумя выборами, каждый из которых меняет значение переменной (например, «доверие»). Проверить, что переменная запоминается и может быть выведена в текст.	контроль результатов творческой деятельности
16			**Поиск**	2	Лицензии: Public Domain, Creative	Каждый участник находит 3	контроль

			графических материалов из открытых источников		Commons (CC0, CC BY и др.), бесплатные для некоммерческого использования. Сайты: OpenGameArt, Pixabay, Unsplash (для фонов), Itch.io (бесплатные ассеты). Правила указания авторства. Ограничения: нельзя брать арты из Pinterest без подтверждённой лицензии.	фона и 2 спрайта персонажа под условно-бесплатной лицензией. Фиксирует ссылки и лицензии в таблице. Групповой просмотр, отбор материалов для своего проекта.	результатов творческой деятельности
17			**Создание графических элементов (спрайтов и фонов)**	2	Основы пиксельной графики и векторной графики. Размеры спрайтов и фонов для новелл (обычно 1920x1080 для фона, спрайт персонажа – 800-1200 пикселей в высоту). Инструменты: GIMP, Krita, Aseprite (пиксель), Inkscape (вектор). Техники создания простого спрайта: силуэт, цвета, тени. Эмоции персонажа (смена выражения лица).	В GIMP или Krita нарисовать простого персонажа (например, рыцаря-манчкина) в одной позе и двумя эмоциями (нейтрально, радостно, зло). Или создать фон комнаты/подземелья. Для начинающих – доработать найденные свободные ассеты (изменить цвет, добавить детали).	контроль результатов творческой деятельности
18			**Сборка прототипа**	2	Объединение сценария, графики, звука в единый работающий продукт. Что такое «прототип»: минимальная версия, демонстрирующая ключевую механику (ветвления, выборы, влияние решений). Итеративная сборка: сначала текст и ветки, потом добавление артов и музыки. Тестирование на себе и других.	Полная сборка прототипа визуальной новеллы на 5-10 минут геймплея (включает: титульный экран, 2-3 сцены, минимум 3 выбора, 2 концовки или значимые изменения в зависимости от выбора). Работа в команде: сценарист, художник, программист.	контроль результатов творческой деятельности
19			**Рефлексия**	2	Понятие рефлексии в проектном обучении. Зачем анализировать свою работу. Типы рефлексии:	Заполнение индивидуального листа рефлексии. Групповое обсуждение: каждый участник	контроль результатов творческой

					эмоциональная, деятельностная, содержательная. Вопросы для самоанализа: что получилось хорошо, что было самым трудным, что бы я сделал иначе, чему я научился.	делится одним успехом и одной ошибкой. Коллективное составление списка рекомендаций для будущих проектов.	деятельности
20			**Защита игры**	2	Как презентовать игру: структура выступления (идея, механики, целевая аудитория, особенности реализации, демонстрация геймплея). Подготовка слайдов (3-5 слайдов) и тизер-видео/скринкаста. Критерии оценки: оригинальность, техническое исполнение, дизайн, соответствие теме, качество презентации.	Публичная защита прототипа перед наставниками и другими группами. Демонстрация игры (5-7 минут). Ответы на вопросы. Просмотр работ других команд, голосование в номинациях («Лучший сюжет», «Лучший визуальный стиль», «Самый манчкинский подход»).	контроль результатов творческой деятельности
Модуль 2: Веб-технологии и программирование для визуальных новелл – 36 часов							
21			**HTML, основные теги (понятие тега, атрибута, базовые элементы: div, p, img, ссылки)**	2	Роль HTML в веб-технологиях. Структура HTML-документа: !DOCTYPE html, html, head, body. Понятия тега и атрибута. Блочные и строчные элементы. Описание базовых тегов: div (контейнер), p (абзац), img (изображение, атрибуты src, alt), a (гиперссылка, атрибут href). Понятие семантической вёрстки.	Создание в VS Code простой HTML-страницы с заголовком, абзацем, изображением и ссылкой. Открытие в браузере. Экспериментирование с атрибутами.	контроль результатов творческой деятельности
22			**Monogatari. Работа с редактором кода (VS Code)**	2	Обзор Monogatari как движка для визуальных новелл, основанного на HTML/CSS/JS. Установка и запуск проекта (через скачивание архива	Скачивание и распаковка стартового шаблона Monogatari. Открытие папки проекта в VS Code. Запуск демо-игры с	контроль результатов творческой деятельности

					или npm). Знакомство с интерфейсом VS Code: обзор папок, открытие проекта, основные горячие клавиши, встроенный терминал. Рекомендуемые плагины для работы с Monogatari (Live Server, Prettier).	помощью Live Server или локального сервера.	
23			**Структура проекта Monogatari — назначение файлов**	2	Обзор корневых папок и файлов: index.html (точка входа), js/script.js (сценарий игры), js/main.js (конфигурация и глобальные настройки), style/main.css (стили), assets/ (изображения, музыка, шрифты). Назначение файла assets/data/assets.json. Краткое знакомство с папкой engine/ (ядро движка, не подлежит изменению).	Открытие каждого из перечисленных файлов, поиск в них знакомых HTML-тегов, CSS-правил, JavaScript-конструкций. Изменение текста в script.js и перезагрузка игры.	контроль результатов творческой деятельности
24			**Визуальное оформление с CSS. Шрифты, свойства шрифтов**	2	Каскадные таблицы стилей (CSS): назначение и способы подключения к HTML. В Monogatari CSS уже подключён. Селекторы (тег, класс, идентификатор). Свойства шрифта: font-family, font-size, font-weight, font-style, color. Подключение внешних шрифтов (Google Fonts) и локальных шрифтов через @font-face. Единицы измерения в CSS: px, em, rem.	В файле style/main.css изменение шрифта текста диалогового окна (например, на Roboto или «Press Start 2P»). Изменение размера и цвета текста.	контроль результатов творческой деятельности
25			**Визуальное оформление с CSS. Фон, цвета**	2	Способы задания цвета в CSS: ключевые слова (red), шестнадцатеричный код (#ffcc00), rgb() / rgba(), hsl(). Свойство background-color. Фоновые	Замена фона главного меню и фоновой заливки диалогового окна. Добавление текстуры или градиента. Создание полупрозрачного фона для окна	контроль результатов творческой деятельности

					изображения: background-image, background-size, background-position, background-repeat. Назначение фона для игрового окна в целом и для отдельных элементов интерфейса.	диалога с использованием rgba.	
26			**Визуальное оформление с CSS. Стилизация окна диалога и кнопок**	2	Определение классов Monogatari, отвечающих за интерфейс (например, .dialog, .text, .choice, .button). Стилизация рамок, теней (box-shadow), скругления углов (border-radius), внутренних и внешних отступов (padding, margin). Псевдоклассы для кнопок: :hover, :active. Оформление кнопок выбора.	Оформление окна диалога в стилистике выбранного сеттинга (например, имитация свитка или панели). Стилизация кнопок выбора (цвет, шрифт, визуальные эффекты при наведении).	контроль результатов творческой деятельности
27			**Визуальное оформление с CSS. Основы анимаций (transition, transform)**	2	Плавные переходы (transition) для изменения свойств элементов (цвета, размера, прозрачности). Трансформации (transform: scale, rotate, translate). Примеры анимации кнопок и появления текста. Анимация с помощью @keyframes. Принципы умеренного использования анимаций в визуальных новеллах.	Стилизация кнопок при наведении курсора (например, изменение масштаба, цвета фона). Анимация появления текста (плавное затухание и сдвиг). Создание пульсирующей иконки перехода.	контроль результатов творческой деятельности
28			**Изображения и звук: добавление фонов, персонажей, музыки в Monogatari**	2	Форматы изображений: PNG (поддержка прозрачности), JPG (для фонов). Размещение файлов в папках assets/scenes/ (фоны) и assets/characters/ (спрайты). Команды show scene и show character в script.js. Настройка позиционирования спрайтов. Рекомендуемые размеры:	Добавление в сцену фона и персонажа из ранее подобранных материалов. Добавление короткой фоновой музыки из бесплатных источников. Проверка корректной загрузки медиа.	контроль результатов творческой деятельности

					фон 1920×1080, спрайт — не более 1500 пикселей по высоте. Форматы звука: MP3, OGG. Папки assets/sounds/ и assets/music/. Команды play music, play sound, stop music. Рекомендации по оптимизации (сжатие изображений через TinyPNG, использование OGG для музыки).		
29			**JavaScript — основы. Данные, типы данных, переменные**	2	Роль JavaScript в Monogatari. Объявление переменных с помощью let и const. Примитивные типы данных: number, string, boolean, null, undefined. Оператор typeof. Преобразование типов. Правила именования переменных (camelCase). Способы хранения данных в Monogatari: monogatari.storage().имяПеременной.	В консоли браузера (F12) объявление нескольких переменных, эксперименты с типами. В файле script.js объявление переменной playerName с присвоением значения и вывод в диалог через monogatari.storage().playerName.	контроль результатов творческой деятельности
30			**JavaScript — основы. Арифметические операции**	2	Арифметические операторы: +, -, *, /, % (остаток от деления), ** (возведение в степень). Приоритет операций. Инкремент (++) и декремент (--). Составное присваивание (+=, -= и др.). Особенности работы с числами с плавающей точкой.	Написание функции или фрагмента кода, вычисляющего суммарный уровень персонажа (сила + ловкость + интеллект) с сохранением результата в переменную. Вывод результата в диалоговое окно.	контроль результатов творческой деятельности
31			**JavaScript — основы. Логика. Условный оператор (if/else)**	2	Логический тип данных (true / false). Операторы сравнения: ==, ===, !=, !==, >, <, >=, <=. Логические операторы && (И), || (ИЛИ), ! (НЕ). Конструкции if, else if, else.	В сценарий новеллы добавление проверки: если значение переменной (например, уровень силы) больше 10, то открывается одна ветка диалога,	контроль результатов творческой деятельности

					Тернарный оператор (? :).	иначе — другая. Использование monogatari.storage().strength.	
32			**JavaScript — основы. Повторение. Циклы (for, while)**	2	Назначение циклов. Синтаксис цикла for (итерация по счётчику). Цикл while (выполнение пока условие истинно). Операторы break и continue. Возможные ошибки (бесконечный цикл). Примеры применения: заполнение массива, перебор элементов.	Написание цикла, который пять раз выводит в консоль различные реплики. В контексте игры — создание цикла для начисления бонусов за несколько последовательных побед над монстрами. Дополнительное задание: вывод списка инвентаря с помощью цикла.	контроль результатов творческой деятельности
33			**JavaScript — основы. Массивы. Объекты**	2	Массивы: способы создания, доступ к элементам по индексу, свойство length. Базовые методы: push, pop, shift, unshift, splice, slice. Перебор массива с помощью цикла for. Объекты: структура «ключ-значение». Создание объекта через {}, доступ к свойствам через точку или квадратные скобки. Вложенные объекты.	Создание массива с именами монстров. Реализация случайного выбора монстра из массива при входе в комнату. Создание объекта player с полями hp, attack, inventory (массив предметов).	контроль результатов творческой деятельности
34			**JavaScript — основы. Объекты HTML (DOM). События**	2	DOM (Document Object Model) — представление HTML-документа в браузере. Поиск элементов с помощью document.getElementById, querySelector. Изменение содержимого и атрибутов элементов. События (click, mouseover, input) и обработчики событий через addEventListener. В Monogatari прямое манипулирование DOM	Создание отдельного HTML-файла с кнопкой и текстовым полем; при нажатии на кнопку текст изменяется. В проекте Monogatari — добавление кастомной кнопки «Показать статистику», которая выводит текущие значения переменных во всплывающее окно.	контроль результатов творческой деятельности

					применяется редко, но может быть полезно для создания кастомных меню.		
35			**Реализация игровой логики средствами JavaScript**	2	Объединение изученных конструкций: переменные, условия, циклы, массивы, объекты. Проектирование игровых систем для «Манчкин-подобной» новеллы: генерация случайного числа (бросок кубика), расчёт бонусов от предметов, проверка уровней, механика «удар в спину». Использование monogatari.storage() для долговременного хранения данных между сценами. Создание вспомогательных функций (например, rollDice()).	Реализация мини-игры «Бой с монстром». Игрок нажимает кнопку «Атаковать», вычисляется случайный урон, уменьшается здоровье монстра. Результат каждого хода выводится в диалоговое окно. При победе выполняется переход к следующей сцене.	контроль результатов творческой деятельности
36			**Сборка прототипа**	2	Требования к прототипу для демонстрации: продолжительность геймплея 5–10 минут, наличие 3–5 значимых выборов, 2 и более концовок или альтернативных путей развития сюжета. Этапы интеграции сценария, графики, звука и игровой логики. Тестирование на наличие ошибок (логические петли, неработающие переходы, неинициализированные переменные). Инструменты отладки: консоль браузера, console.log. Экспорт игры для демонстрации (локальный сервер, сборка в	Доработка командами своих прототипов по предоставленному чек-листу. Проведение тестирования силами соседней команды. Исправление критических ошибок. Подготовка демонстрационной версии.	контроль результатов творческой деятельности

					исполняемый файл через Electron).		
37			**Рефлексия (анализ результатов и процесса работы)**	2	Понятие рефлексии в проектном обучении. Анализ сильных и слабых сторон реализованного проекта, а также командного взаимодействия. Выявление наиболее сложных с точки зрения программирования этапов. Определение возможных улучшений архитектуры игры.	Заполнение рефлексивной карты по трём колонкам: «достижения», «проблемы», «идеи для улучшения». Групповое обсуждение: каждая команда делится одним техническим успехом и одной ошибкой. Коллективная выработка рекомендаций (типичные ошибки и способы их избегания).	контроль результатов творческой деятельности
38			**Защита игры (презентация перед аудиторией)**	2	Структура презентации проекта: название, жанр, основная идея, целевая аудитория, описание ключевых особенностей реализации (использованные возможности HTML/CSS/JS). Демонстрация геймплея продолжительностью 3–5 минут с показом важнейших механик. Критерии оценки: техническая сложность, стабильность работы, дизайн интерфейса, оригинальность, качество устной презентации. Рекомендации по публичному выступлению.	Публичная защита прототипа перед наставниками и другими учебными группами. Каждая команда демонстрирует игру (5–7 минут) и отвечает на вопросы. Проведение зрительского голосования в номинациях.	контроль результатов творческой деятельности
Модуль 3: Классическое программирование на Python. Основы алгоритмов – 36 часов							
39			**Язык программирования Python. Место Python в семье**	2	История создания Python, философия языка (PEP 20 – The Zen of Python). Классификация языков программирования: компилируемые	Установка интерпретатора Python (последней стабильной версии) и среды разработки (VS Code или PyCharm Community	контроль результатов творческой деятельности

			языков		и интерпретируемые, высокоуровневые и низкоуровневые. Место Python среди других языков (C++, Java, JavaScript). Области применения Python: веб-разработка, анализ данных, машинное обучение, автоматизация, научные вычисления, образование. Преимущества и недостатки Python.	Edition). Написание первой программы: вывод строки «Hello, World!» в консоль. Запуск скрипта из командной строки и из IDE.	
40			**Python — основы. Данные, типы данных, переменные**	2	Понятие переменной в программировании. Правила именования переменных (PEP 8). Присваивание значения. Базовые типы данных: целые числа (int), числа с плавающей точкой (float), строки (str), булевый тип (bool). Определение типа переменной с помощью type(). Динамическая типизация в Python. Преобразование типов: int(), float(), str(). Консольный ввод и вывод: print() и input().	Написание скрипта, который запрашивает у пользователя имя и возраст, затем выводит приветствие и тип введённых данных. Эксперименты с преобразованием типов.	контроль результатов творческой деятельности
41			**Python — основы. Арифметика**	2	Арифметические операторы: сложение (+), вычитание (-), умножение (*), деление (/), целочисленное деление (//), остаток от деления (%), возведение в степень (**). Приоритет операций, использование круглых скобок. Особенности арифметики с числами с плавающей точкой (погрешности вычислений). Модуль math и его основные функции (math.sqrt,	Написание программы для вычисления площади круга, дискриминанта квадратного уравнения, среднего арифметического нескольких чисел. Использование math для более сложных вычислений.	контроль результатов творческой деятельности

					math.floor, math.ceil).		
42			**Python — основы. Логика. Условный оператор (if/else)**	2	Логический тип bool, значения True и False. Операторы сравнения: ==, !=, >, <, >=, <=. Логические операторы: and, or, not. Конструкции ветвления: if, elif, else. Вложенные условные операторы. Тернарный условный оператор (выражение X if условие else Y).	Написание программы, определяющей, является ли год високосным. Программа для сравнения трёх чисел и нахождения наибольшего. Реализация простого калькулятора с выбором действия через if/elif/else.	контроль результатов творческой деятельности
43			**Python — основы. Повторение. Циклы (for, while)**	2	Цикл while: синтаксис, условие продолжения, опасность бесконечного цикла. Операторы break и continue. Цикл for: итерация по последовательностям (строки, списки, диапазоны). Функция range(): различные варианты вызова (range(stop), range(start, stop), range(start, stop, step)). Вложенные циклы.	Написание программы для вычисления факториала числа (через for и через while). Программа для вывода таблицы умножения. Алгоритм проверки, является ли число простым.	контроль результатов творческой деятельности
44			**Python — основы. Списки, кортежи, множества, словари**	2	Списки (list): создание, доступ по индексу, срезы ([start:stop:step]), основные методы (append, extend, insert, remove, pop, index, count, sort, reverse). Вложенные списки. Кортежи (tuple): неизменяемость, создание, доступ по индексу, методы (count, index). Множества (set): неупорядоченность, уникальность элементов, операции над множествами (объединение, пересечение, разность, симметрическая разность), методы	Реализация программы для подсчёта частоты слов в предложении (с использованием словаря). Работа со списком студентов и их оценками (средний балл). Использование множества для удаления дубликатов из списка.	контроль результатов творческой деятельности

					(add, remove, discard, clear). Словари (dict): пары «ключ–значение», создание, доступ по ключу, методы (keys, values, items, get, pop, update).		
45			**Python — основы. Стиль написания программ. Идиомы Python**	2	Рекомендации PEP 8: отступы, максимальная длина строки, пробелы вокруг операторов, именование переменных и функций (snake_case), констант (UPPER_CASE). Идиомы Python (Pythonic code): распаковка последовательностей (a, b = b, a), генераторы списков и словарей, использование enumerate() и zip(), конструкция with для работы с файлами. Документирование кода с помощью docstring.	Рефакторинг «некрасивого» кода в соответствии с PEP 8 и идиомами. Написание функции с docstring. Применение генератора списков для создания квадратов чисел. Использование enumerate в цикле для вывода элементов списка с индексами.	контроль результатов творческой деятельности
46			**Алгоритмы. Сложность алгоритма**	2	Понятие алгоритма, его свойства (конечность, детерминированность, массовость). Необходимость анализа эффективности. Временная и пространственная сложность. «О-большое» (Big O notation): определение, основные классы сложности: O(1) – константная, O(log n) – логарифмическая, O(n) – линейная, O(n log n) – линейно-логарифмическая, $O(n^2)$ – квадратичная, $O(2^n)$ – экспоненциальная. Примеры алгоритмов для каждого класса.	Анализ простых фрагментов кода (циклы, вложенные циклы) с определением их сложности. Сравнение времени выполнения двух алгоритмов на разных объёмах входных данных (экспериментально с помощью модуля time).	контроль результатов творческой деятельности
47			**Алгоритмы.**	2	Линейный поиск: принцип работы,	Написание функции линейного	контроль

			Базовый поиск		сложность O(n). Реализация на Python. Поиск минимального/максимального элемента. Бинарный поиск: условие применимости (отсортированный массив), алгоритм «деления пополам», сложность O(log n). Итеративная и рекурсивная реализации.	поиска для нахождения индекса заданного элемента. Реализация бинарного поиска на отсортированном списке. Сравнение времени работы обоих алгоритмов на больших массивах.	результатов творческой деятельности
48			**Алгоритмы. Простые сортировки**	2	Задача сортировки. Классификация алгоритмов сортировки: устойчивые и неустойчивые, «на месте» и требующие дополнительной памяти. Сортировка пузырьком (Bubble sort): описание, сложность $O(n^2)$, оптимизация с флагом обмена. Сортировка вставками (Insertion sort): описание, эффективна на почти отсортированных данных, сложность $O(n^2)$ в худшем случае. Сортировка выбором (Selection sort): поиск минимума и перемещение в начало.	Реализация каждого из трёх алгоритмов сортировки. Тестирование на случайных списках разного размера, измерение времени выполнения. Визуализация процесса сортировки (вывод промежуточных состояний).	контроль результатов творческой деятельности
49			**Алгоритмы. Рекурсия и «разделяй и властвуй»**	2	Понятие рекурсии: базовый случай и рекурсивный шаг. Рекурсивные функции на Python. Риски: глубина рекурсии, переполнение стека (sys.setrecursionlimit). Классические примеры: вычисление факториала, чисел Фибоначчи (наивная рекурсия и её недостатки). Парадигма «разделяй и властвуй» (Divide and Conquer). Алгоритм сортировки	Написание рекурсивной функции для вычисления факториала и чисел Фибоначчи. Реализация сортировки слиянием. Сравнение времени работы сортировки слиянием с простыми сортировками на больших данных.	контроль результатов творческой деятельности

					слиянием (Merge sort): описание, разделение на подмассивы, слияние, сложность O(n log n).		
50			**Алгоритмы. Алгоритмы на структурах данных**	2	Стек (Stack): LIFO, реализация на Python с использованием списка (append, pop). Примеры применения: проверка сбалансированности скобок, обратная польская запись. Очередь (Queue): FIFO, использование collections.deque для эффективной работы с обоими концами. Связный список: односвязный и двусвязный (теоретическое понятие). Хеш-таблица: принцип работы, разрешение коллизий. В Python это словарь (dict).	Реализация проверки баланса скобок с помощью стека. Реализация очереди для задачи обработки заявок. Использование deque для организации очереди.	контроль результатов творческой деятельности
51			**Алгоритмы. Обход графов**	2	Основные понятия теории графов: вершины, рёбра, ориентированные и неориентированные графы, взвешенные графы. Способы представления графов в программе: матрица смежности, список смежности (словарь списков). Поиск в глубину (DFS): рекурсивная и итеративная реализации (стек). Поиск в ширину (BFS): использование очереди. Нахождение кратчайшего пути в невзвешенном графе.	Реализация графа с использованием списка смежности. Написание DFS для обхода всех вершин. Реализация BFS для нахождения кратчайшего расстояния между двумя вершинами.	контроль результатов творческой деятельности
52			**Алгоритмы. Динамическое**	2	Идея динамического программирования: разбиение	Реализация чисел Фибоначчи с мемоизацией и табуляцией.	контроль результатов

			программирование (база)		задачи на подзадачи, запоминание результатов для повторного использования (мемоизация). Сравнение с наивной рекурсией на примере чисел Фибоначчи. Принципы оптимальности для подзадач. Мемоизация «сверху вниз»: рекурсия с кэшированием (functools.lru_cache). Табуляция «снизу вверх»: заполнение массива (DP-таблицы).	Решение классической задачи «Кубики» (найти количество способов добраться до вершины лестницы) или задачи о наборе монет (минимальное количество монет для сдачи).	творческой деятельности
53			**Решение задач базового уровня сложности**	2	Обзор типов задач, которые считаются базовыми: работа с числами (простые числа, делители, палиндромы), строки (анаграммы, регистр, удаление символов), списки (сумма элементов, поиск дубликатов, срезы). Формулировка требований к решению: входные данные, выходные данные, ограничения.	Решение 4–5 задач из сборников (например, с сайта Codewars уровень 8 kyu или 7 kyu, или с acmp.ru номера 1–50). Задачи разбираются индивидуально или в парах. Проводится обсуждение различных подходов.	контроль результатов творческой деятельности
54			**Решение задач повышенного уровня сложности**	2	Признаки задач повышенной сложности: комбинирование нескольких структур данных, необходимость выбора эффективного алгоритма (не квадратичного), задачи на рекурсию и перебор, обработка больших входных данных.	Решение 3–4 задач из олимпиадных источников (например, Codeforces рейтинг 1000-1200, acmp.ru номера 100–300). Примеры: задача на поиск подстроки, задача на динамическое программирование начального уровня, задача на работу с графами (поиск компонент связности).	контроль результатов творческой деятельности
55			**Решение задач**	2	Особенности олимпиадных задач:	Разбор и решение 2–3 задач	контроль

			олимпиадного уровня сложности		нестандартные условия, многомерные структуры данных, оптимизация по времени и памяти, применение изученных алгоритмов (бинарный поиск, ДП, BFS/DFS, рекурсия). Понятие «жадные алгоритмы» (краткое введение).	уровня всероссийской олимпиады (не самые сложные варианты). Примеры: задача на поиск кратчайшего пути во взвешенном графе (алгоритм Дейкстры — ознакомительно), задача на динамическое программирование на двумерном массиве («Черепашка»). Реализация в парах с последующей защитой решения.	результатов творческой деятельности
56			**Рефлексия**	2	Анализ прогресса в изучении Python и алгоритмов. Какие темы были наиболее сложными? Какие алгоритмы требуют дополнительного повторения? Оценка собственного уровня: от написания простых скриптов до решения олимпиадных задач. Планирование дальнейшего обучения: ресурсы (Codeforces, LeetCode, Stepik, «Типичный программист»), геймджемы, хакатоны.	Заполнение индивидуальной рефлексивной карты (достижения, пробелы, цели). Коллективное обсуждение: обмен опытом решения задач. Создание персональной дорожной карты «Куда двигаться после курса».	контроль результатов творческой деятельности
Модуль 4: Разработка проекта с использованием искусственного интеллекта – 32 часа							
57			**Что такое проект**	2	Определение понятия «проект» в контексте IT-разработки. Отличительные признаки проекта: целеполагание, ограниченность во времени, уникальность результата, ресурсные ограничения. Типы	Анализ 2–3 примеров проектов (из материалов Кванториума или открытых источников) на предмет наличия признаков проекта. Групповое обсуждение: чем проект	

					проектов по сложности, продолжительности и составу команды. Примеры успешных проектов в области создания игр и приложений.	отличается от «просто написания кода».	
58			**Идея проекта**	2	Источники идей: личный опыт, наблюдение, потребности рынка, анализ существующих решений. Методы генерации идей: мозговой штурм, майндмэппинг, метод «проблема – решение». Критерии отбора идей: новизна, реализуемость, полезность, соответствие техническим навыкам команды.	В командах (2–3 человека) проведение мозгового штурма для генерации не менее 5 идей проектов. Выбор одной идеи для дальнейшей разработки. Краткое устное обоснование выбора.	
59			**Аудит цели проекта**	2	Понятие цели проекта. Техника SMART: цель должна быть конкретной, измеримой, достижимой, релевантной и ограниченной по времени. Отличие цели от задачи. Аудит цели: проверка на реалистичность, оценка ресурсов, выявление рисков.	Формулировка цели проекта по SMART. Проведение аудита цели внутри команды (каждый проверяет цель соседней команды). Корректировка цели по результатам аудита.	
60			**Предварительное техническое задание (ТЗ)**	2	Назначение технического задания. Структура предварительного ТЗ: назначение и цели, функциональные требования, нефункциональные требования (производительность, безопасность, платформа), ограничения и допущения. Пример ТЗ для небольшого программного продукта.	Составление предварительного технического задания на выбранный проект (объёмом 1–2 страницы). Включение требований к пользовательскому интерфейсу, к данным, к среде исполнения.	

61			**Роли в проекте. Распределение ролей в тандеме «человек – ИИ»**	2	Классические роли в IT-проекте: менеджер, аналитик, разработчик, тестировщик, документатор. Особенности работы с инструментами искусственного интеллекта (ChatGPT, GitHub Copilot, Midjourney и др.) как с «виртуальными помощниками». Распределение ответственности: человек ставит задачу, контролирует результат, принимает итоговые решения; ИИ генерирует варианты, предлагает решения, выполняет рутинные операции.	Определение ролей внутри команды (какой участник за что отвечает). Составление схемы «Кто принимает решения по каким вопросам с участием ИИ». Назначение одного участника ответственным за работу с ИИ-инструментами.	
62			**Итеративная разработка**	2	Понятие итерации (цикла) в разработке. Сравнение каскадной (водопадной) и итеративной моделей. Преимущества итеративного подхода: быстрая обратная связь, снижение рисков, возможность изменять требования. Длина итерации (спринта) в учебном проекте (3–5 дней).	Планирование итераций для своего проекта: определение минимального набора функций для первой итерации, второй и т.д. Составление графика итераций с указанием дат завершения каждой.	
63			**Внутренние тесты и разработка через тестирование (TDD)**	2	Понятие тестирования программного обеспечения. Внутренние тесты (силами разработчиков). Разработка через тестирование (Test-Driven Development, TDD): цикл «красный – зелёный – рефакторинг». Написание простых модульных тестов (unit tests) для Python (библиотека unittest или pytest).	Выбор одного модуля проекта (например, функция расчёта бонуса в игре). Написание теста, затем реализация функции, проходящей тест. Ознакомление с запуском тестов и анализом результатов.	

64			**Прототип. Минимальный работоспособный продукт (MVP)**	2	Определение прототипа и минимального работоспособного продукта (Minimum Viable Product, MVP). Отличия: прототип может не иметь полной функциональности, но демонстрирует ключевую идею; MVP – это первая рабочая версия, которую можно показывать пользователям. Критерии MVP для учебного проекта.	Определение состава MVP для своего проекта (какие функции обязательно должны быть реализованы к концу первой итерации). Создание прототипа (бумажного или цифрового) интерфейса или ключевой механики.	
65			**Презентация прототипа**	2	Цели презентации прототипа: получение обратной связи, проверка жизнеспособности идеи, уточнение требований. Структура презентации: проблема, решение, демонстрация ключевого экрана или сценария, вопросы к аудитории. Рекомендации по проведению (хронометраж 5 минут на команду).	Публичная презентация прототипов перед наставником и другими командами. Сбор обратной связи (письменно или устно). Обсуждение полученных замечаний и предложений.	
66			**Добавление функционала в проект**	2	Принципы добавления новых функций после создания MVP. Приоритизация: метод MoSCoW (must have, should have, could have, won't have). Управление изменениями: ведение списка задач (backlog), оценка трудозатрат. Использование ИИ для генерации кода новых функций.	Формирование backlog на 2–3 следующих итерации. Реализация одной дополнительной функции с использованием ИИ-ассистента (например, ChatGPT запрашивается фрагмент кода или алгоритм). Интеграция функции в проект.	
67			**Интеграционное тестирование**	2	Отличие интеграционного тестирования от модульного. Проверка взаимодействия между компонентами проекта. Типичные	Проведение интеграционного тестирования созданного проекта (тестирование целых сценариев: запуск игры,	

					проблемы при интеграции: несовместимость интерфейсов, ошибки передачи данных, сбои в работе связанных функций. Подходы к интеграционному тестированию: «снизу вверх», «сверху вниз», «большой взрыв».	выполнение цепочки действий, достижение конца). Фиксация обнаруженных ошибок и их исправление.	
68			**Презентация продуктового результата проекта**	2	Отличия презентации продуктового результата от презентации прототипа. Акцент на завершённости, стабильности, пользовательской ценности. Структура: введение, демонстрация готового продукта, ключевые технические решения, полученные результаты и метрики. Подготовка демонстрационной среды (установка на ноутбук, запись видео).	Репетиция презентации в команде. Публичная демонстрация готового продукта (игра, приложение, веб-сервис) перед аудиторией. Ответы на вопросы.	
69			**Документирование проекта**	2	Виды документации: пользовательская (инструкция по установке и использованию), техническая (описание архитектуры, API, базы данных), проектная (ТЗ, план итераций, результаты тестирования). Требования к документации: полнота, понятность, актуальность. Использование ИИ для помощи в составлении документации.	Составление двух документов: «Руководство пользователя» (1 страница) и «Краткое техническое описание» (1–2 страницы). Включение скриншотов и примеров кода.	
70			**Рефлексия полученного опыта**	2	Анализ процесса разработки с использованием ИИ. Какие задачи были эффективно решены с	Заполнение индивидуального опросника рефлексии по образцу (5 вопросов: что	

					помощью ИИ, а какие потребовали вмешательства человека. Оценка продуктивности работы в тандеме. Выявление приобретённых навыков: формулирование запросов (промтов) для ИИ, критическая оценка результатов, интеграция сгенерированного кода.	получилось, что нет, чему научился, что было неожиданным, как изменилось отношение к ИИ). Групповое обсуждение успехов и трудностей.	
71			**Защита проекта (презентация перед аудиторией)**	2	Финальная защита проекта – итоговое мероприятие курса. Структура выступления: проблема, решение, технические детали, демонстрация (видео или живая), вклад ИИ в разработку, планы развития. Критерии оценки: полнота реализации ТЗ, качество кода и документации, умение отвечать на вопросы, использование ИИ.	Публичная защита проекта перед комиссией (наставники, приглашённые эксперты, другие группы). Длительность защиты – 7–10 минут на команду, включая демонстрацию и вопросы.	
72			**Обобщение знаний, полученных по итогам курса**	2	Связь всех модулей (проектирование новеллы, веб-технологии, классическое программирование, разработка с ИИ). Формирование целостной картины разработки программного продукта. Вопросы для итогового собеседования: жизненный цикл проекта, роль документации, виды тестирования, использование ИИ.	Проведение итоговой тестовой работы (20 вопросов с выбором ответа) или устного собеседования. Подведение итогов курса, вручение сертификатов/дипломов.	
			ИТОГО	144			

Календарно-тематический план
к дополнительной общеразвивающей программе
«IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»
2 год обучения

	Дата проведения занятия		Название раздела, темы	Количество часов	Содержание деятельности		Форма аттестации
	по плану	по факту			Теоретическая часть занятия	Практическая часть занятия	
Модуль 1. Командная разработка и управление IT-проектами – 36 часов							
1.			**Введение в командную разработку. Жизненный цикл IT-проекта**	2	Понятие командной разработки, её отличие от индивидуальной. Преимущества и вызовы работы в команде. Жизненный цикл программного продукта: анализ → проектирование → разработка → тестирование → внедрение → сопровождение. Модели жизненного цикла: каскадная, итеративная, Agile (обзор).	Дискуссия «Почему большие проекты невозможно создать в одиночку?». Составление схемы жизненного цикла для вымышленного IT-проекта (например, мобильного приложения).	Устный опрос, проверка схемы
2.			**Роли в команде: продюсер, PM, разработчик, тестировщик, дизайнер. Распределение ролей**	2	Описание ключевых ролей в IT-команде: продюсер (видение продукта), проектный менеджер (процессы), разработчик (код), тестировщик (качество), дизайнер (интерфейс и опыт). Дополнительные роли: аналитик, DevOps, документатор. Ответственность и зоны взаимодействия.	Упражнение «Собери команду»: для заданного проекта (например, создание визуальной новеллы) участники распределяют роли внутри мини-групп, аргументируют выбор. Составление матрицы ответственности.	Контроль результатов творческой деятельности (схема распределения ролей)

3.			**Системы управления проектами. Создание доски задач**	2	Назначение систем управления проектами. Обзор «Битрикс24» (карточки, списки, метки), использование совместных электронных таблиц для планирования. Выбор инструмента под размер команды. Базовые элементы: задачи (карточки), статусы, исполнители, дедлайны.	Регистрация в выбранной системе. Создание доски для учебного проекта: колонки «To do», «In progress», «Done». Добавление 3–4 задач, назначение ответственных.	Проверка созданной доски, скриншот или ссылка	3
4.			**Командная работа: мозговой штурм, определение идеи сквозного проекта. Сервис Яндекс.Доски**	2	Методы генерации идей: мозговой штурм (правила, фазы), метод SCAMPER, майндмэппинг. Критерии выбора идеи для сквозного проекта: реализуемость за спринты, наличие технического задела, интерес команды.	Проведение мозгового штурма в командах (3–4 человека), фиксация всех идей на доске (электронной или физической). Выбор одной идеи голосованием. Формулировка названия и краткого описания проекта.	Наблюдение, утверждение идеи наставником	4
5.			**Agile и Scrum: спринты, бэклог, ежедневные встречи**	2	Принципы Agile (гибкость, итеративность, обратная связь). Scrum: роли (Scrum-мастер, владелец продукта, команда), артефакты (бэклог, спринт-бэклог, инкремент), события (планирование спринта, ежедневный стендап, ревью, ретроспектива). Длительность спринта (1–2 недели). Бэклог продукта – список всех желаемых функций. Приоритизация: метод MoSCoW, матрица усиление/ценность. Планирование спринта: отбор задач из бэклога, разбиение на подзадачи, оценка трудоёмкости (story points или часы).	Моделирование спринта: команды формируют недельный бэклог из 5–7 задач, выбирают задачи на спринт (2 дня или 1 неделя). Проведение короткого стендапа (имитация). Асинхронный daily standup. Фиксирование итогов коротким сообщением в командном чате. Использование шаблонов или стикеров для заполнения информации заранее.	Контроль результатов творческой деятельности (бэклог и план спринта)	

6.			**Git: установка, базовые команды (init, add, commit, status, log). Ветки (branch, checkout, merge), разрешение конфликтов**	2	Система контроля версий Git: назначение, основные понятия (репозиторий, коммит, ветка). Установка Git, настройка имени пользователя. Команды: git init, git add, git commit, git status, git log. Ветвление: git branch, git checkout, git merge. Понятие конфликта и его разрешение (ручное редактирование).	Установка Git (если не установлен). Создание локального репозитория, добавление файла, коммит. Создание ветки, переключение, слияние. Искусственное создание конфликта и его разрешение.	Контроль результатов творческой деятельности (скриншоты выполненных команд, репозиторий на локальной машине)	6
7.			**Разработка спринта 1: распределение задач, работа в ветках Git. Ежедневные стендапы (имитация + дистант). Ревью кода внутри команды**	2	Как организовать работу в спринте: назначение исполнителей, создание веток для каждой задачи, регулярные коммиты. Ежедневный стендап (что сделано, что планирую, какие блокеры). Ревью кода: цели, чек-лист, конструктивная обратная связь.	Команды выполняют задачи первого спринта (программирование, вёрстка, дизайн – в зависимости от проекта). Проводят 2–3 коротких стендапа (один – очно, остальные – в чате). Организуют ревью кода через Pull request.	Контроль результатов творческой деятельности (выполненные задачи, активность в репозитории, участие в ревью)	7
8.			**Работа с удалённым репозиторием: Pull request и code review. Баг-трекинг: оформление issue, метки, приоритеты**	2	Удалённые репозитории (GitVerse, Mos.hub). Команды git remote, git push, git pull, git clone. Процесс Pull request: форк, ветка, запрос на слияние, обсуждение, ревью кода. Баг-трекинг: задачи (issues), метки (bug, enhancement, question), приоритеты (high, medium, low).	Регистрация на GitVerse, создание удалённого репозитория. Отправка локальных коммитов, клонирование. Создание ветки на GitVerse, открытие Pull request. Добавление issue к репозиторию с меткой и приоритетом.	Контроль результатов творческой деятельности (ссылка на репозиторий, активный PR и issue)	8
9.			**Разработка спринта 2:**	2	Продолжение работы: добавление новых функций. Конфликты при	Команды выполняют задачи второго спринта. При	Контроль результатов	9

			добавление функционала, решение конфликтов. Тестирование и приёмка: оформление багов, приоритизация		слиянии (git merge) – причины и способы разрешения. Тестирование: виды тестов (модульное, интеграционное, ручное). Оформление багов: воспроизведение, ожидаемое/фактическое поведение, серьёзность и приоритет.	возникновении конфликтов в Git – разрешают их. Проводят ручное тестирование друг у друга (cross-testing). Оформляют баги в issue с метками и приоритетами.	творческой деятельности (завершённые задачи спринта, исправленные баги, разрешённые конфликты)	
10.			**CI/CD: знакомство (GitVerse). Автоматическая сборка и тесты**	2	Понятие CI/CD (непрерывная интеграция и непрерывная доставка). Зачем автоматизировать сборку, тестирование, деплой. Примеры: GitHub Actions, GitLab CI, Jenkins. Базовые понятия: workflow, job, step, runner, конфигурационный файл (YAML).	Создание простого workflow в GitVerse для своего репозитория: запуск проверки синтаксиса (например, для Python или Markdown) при каждом push. Просмотр результатов в интерфейсе GitHub.	Контроль результатов творческой деятельности	10
11.			**Разработка спринта 3: доработка по фидбеку. Интеграционное тестирование и финальная сборка**	2	Учёт обратной связи: приоритизация изменений и добавление в бэклог. Интеграционное тестирование – проверка взаимодействия всех компонентов. Финальная сборка: компиляция, создание исполняемого файла, подготовка дистрибутива или деплой на хостинг.	Команды вносят изменения, запланированные на спринт 3 (по обратной связи). Проводят интеграционное тестирование (прогоняют сценарий от начала до конца). Выполняют финальную сборку своего сквозного проекта (если игра – экспорт, если веб-приложение – деплой).	Контроль результатов творческой деятельности (работоспособная финальная сборка, отчёт о тестировании)	11
12.			**Техническая документация: README, Wiki, архитектурные схемы**	2	Зачем нужна документация. Файл README (назначение, структура: название, описание, установка, использование, лицензия). Wiki как расширенная документация. Архитектурные схемы: диаграммы	Создание README для учебного проекта в формате Markdown. Построение простой архитектурной схемы (например, «пользователь → веб-интерфейс → сервер → база	Контроль результатов творческой деятельности (файл README в	12

					потоков данных, диаграммы классов, компонентные диаграммы. Инструмент - draw.io.	данных») в draw.io.	репозитории, схема в формате изображения)	
13.			**Демо прототипа. Сбор обратной связи. Ретроспектива**	2	Цель демо-встречи (спринт-ревью) – показать результат заинтересованным лицам. Как собирать обратную связь (опросы, устные комментарии, фиксация пожеланий). Ретроспектива: анализ того, что пошло хорошо, что плохо, что улучшить. Техники «Start, Stop, Continue» или «Море, якоря, облака».	Демонстрация работающего продукта (промежуточный результат). Получение обратной связи. Проведение ретроспективы внутри команды с заполнением таблицы «Что улучшить».	Защита этапа проекта (демо спринта), ретроспективная карта	13
14.			**Внесение изменений в проект на основании обратной связи. Рефакторинг кода. Код-ревью. Обсуждение Pull Request в GitVerse. Pinning tests**	2	Обратная связь как источник улучшений: сбор, систематизация, приоритизация (критические, желательные). Рефакторинг – изменение внутренней структуры без изменения поведения (устранение дублирования, длинных функций, магических чисел). Код-ревью: цели, чек-лист ревьювера, правила конструктивной обратной связи. Pull Request в GitVerse: создание, описание, запрос ревью, обсуждение комментариев, доработка, слияние. Pinning tests (тесты на регрессию) – фиксирующие тесты, которые проверяют, что исправленная ошибка не вернётся (сначала тест падает, после исправления – проходит).	Команды получают обратную связь по сквозному проекту, составляют список изменений. Каждый участник выполняет рефакторинг одного фрагмента кода. В парах проводят код-ревью: один создаёт Pull Request с изменениями, другой оставляет комментарии, автор дорабатывает и мержит. Для исправленного бага пишут тест (в CI или простой assert), демонстрируя, что он падает до правки и проходит после.	Контроль результатов творческой деятельности (оформленный Pull Request с ревью и мержем, добавленный тест, отчёт о рефакторинге)	14
15.			**AI-помощь в**	2	AI-анализ кода: выявление	Анализ текущей кодовой базы	Контроль	15

			модификации кода		антипаттернов, мёртвого кода, логических ошибок. Комментирование: фиксация намерения, инвариантов, предусловий. Точки модификации по ТЗ: семантическое сопоставление требований с модулями, классификация (добавить/изменить/удалить). Генерация изменений: фрагменты кода с обоснованием выбора алгоритма. Проверка: синтаксис (AST, линтеры), логика (граничные случаи), генерация тестов. Рефакторинг: единый стиль, устранение дублирования, переименование. Отчёт: журнал правок, обновлённая документация, риски.	при помощи AI, комментирование, и выявление точек модификации согласно техническому заданию; генерация конкретных изменений (фрагменты кода, правки структур данных, оптимизация алгоритмов) с пояснением логики; проверка изменений на синтаксические и логические ошибки, предложение тестовых сценариев; рефакторинг и унификация стиля для поддержки сопровождаемости; документирование внесённых правок и формирование итогового отчёта.	результатов творческой деятельности (оформленный Pull Request с ревью и мержем, добавленный тест, отчёт о рефакторинге)	
16.			**Презентация готового продукта**	2	Подготовка к презентации: структура (проблема, решение, демонстрация, технологии, дальнейшие планы). Правила публичного выступления: контакт с аудиторией, управление временем, ответы на вопросы. Роль каждого члена команды в презентации.	Репетиция презентации в команде. Публичная презентация. Ответы на вопросы.	Защита проекта	16
17.			**Итоговая аттестация по модулю: защита портфолио (доска задач, репозиторий, документация)**	2	Что входит в портфолио: ссылка на репозиторий GitVerse, доска задач, README, пользовательская документация, архитектурные схемы, отчёты о тестировании, ретроспективные карты. Критерии оценки полноты и качества портфолио.	Оформление портфолио в едином документе. Индивидуальное или командное собеседование с наставником: защита портфолио, обоснование принятых решений.	Защита портфолио (оценка наставника, самооценка команды)	17

18.			**Рефлексия образовательных результатов**	2	Значение рефлексии в проектном обучении. Вопросы для самоанализа: какие навыки командной работы приобрели, что было самым сложным, какой вклад внёс каждый участник, как изменилось понимание IT-проектов. Оценка своей роли и достижений.	Заполнение индивидуального рефлексивного листа (например, по технике «Плюс-минус-интересно»). Групповое обсуждение: обмен впечатлениями о модуле, советы будущим командам.	Беседа, наблюдение, письменная рефлекси	18
Модуль 2. Событийное программирование в среде «Берлога» + основы 3D-моделирования – 36 часов								
19.			**Знакомство с игрой «Берлога: Защита пасеки». Разбор структуры готового игрового проекта**	2	Обзор среды «Берлога»: назначение, интерфейс, основные возможности. Запуск готового проекта «Защита пасеки». Определение ключевых игровых объектов (медведь, пчёлы, улей, пасечник). Общее представление о логике: что происходит при касании, как начисляются очки, условия проигрыша/выигрыша.	Запуск проекта, игра в режиме тестирования. Исследование структуры проекта: список объектов, настройки сцены, блоки поведения. Заполнение таблицы «Объект – его роль – пример действия».	Беседа, наблюдение; таблица описания объектов	
20.			**Изучение логики поведения объектов. Создание схемы игрового процесса с выделением ключевых элементов логики**	2	Понятие «логика поведения»: как объекты реагируют на действия игрока и друг на друга. Составление блок-схемы игрового процесса (старт → движение → столкновение → изменение счётчика → окончание).	Разбор логики двух-трёх объектов в «Защите пасеки». Создание графической схемы игрового процесса с выделением условий и переходов. Презентация схемы в группе.	Контроль результатов творческой деятельности (схема игрового процесса)	
21.			**Машины состояний. Понятие состояния. Создание системы с простыми**	2	Что такое состояние объекта (например, «идёт», «стоит», «атакует»). Конечный автомат (Finite State Machine) как модель поведения. Примеры из жизни: светофор, лифт. Зачем нужны состояния в играх	В среде «Берлога» создание двух состояний для простого объекта (например, для пчелы: «летает» и «отдыхает»). Реализация переключения по таймеру. Проверка работы.	Контроль результатов творческой деятельности (проект с двумя состояниями)	

			состояниями		(упрощение логики, наглядность).		
22.			**Машины состояний. Переходы. Создание схемы машины состояний для персонажа**	2	Понятие перехода (транзиции). Условия перехода (по времени, по событию, по значению переменной). Способы задания переходов в «Берлоге» (блоки «если», события столкновения и т.п.).	Выбор персонажа (медведь). Проектирование схемы его состояний: «спит», «идёт к улью», «атакует», «убегает». Прорисовка переходов между состояниями. Реализация двух переходов в проекте (например, по таймеру и по касанию).	Контроль результатов творческой деятельности (схема + фрагмент проекта)
23.			**Машины состояний. События. Обработка событий в игровой логике. Роль триггеров и условий**	2	События (клик, касание, столкновение, получение сообщения). Триггеры как активаторы переходов. Условия (проверка переменных). Разница между событием и условием: событие происходит (единоразово), условие может длиться.	Добавление в проект события «при касании медведя с ульем» – переход в состояние «атака». Добавление условия «если здоровье улья < 0» – переход в состояние «разрушен». Реализация в «Берлоге».	Контроль результатов творческой деятельности (работающая реакция на события)
24.			**Реализация логики смены состояний на практике (в среде «Берлога»)**	2	Повторение и обобщение: цикл состояний, обработка ошибок (зависание в состоянии). Рекомендации по именованию состояний и переменных.	Командная работа: каждый участник создаёт мини-сцену с двумя персонажами (например, кот и мышь), у каждого не менее трёх состояний с переходами по событиям. Взаимное тестирование (один играет, другой смотрит на корректность смены состояний).	Контроль результатов творческой деятельности (проект с полной машиной состояний)
25.			**Турнир по программированию. Виды конфликтов**	2	Что такое конфликт в нарративе и игровом дизайне. Глобальный конфликт (война, вторжение, катастрофа) и локальный (спор	Дискуссия «Примеры конфликтов в известных играх». Работа в парах: придумать по одному глобальному и	Наблюдение, устные ответы

			(глобальный, локальный). Роль конфликта в развитии истории		персонажей, внутренняя борьба). Роль конфликта в мотивации игрока и смене состояний.	локальному конфликту, которые можно реализовать в «Берлоге» (например, защита пасеки – глобальный; конкуренция двух пчёл за мёд – локальный).	
26.			**Написание короткой сцены, в которой проявляется конфликт (реализация в «Берлоге»)**	2	Как перенести конфликт в игровую механику: конфликт как условие смены состояний, как цель уровня. Структура короткой сцены (завязка, обострение, разрешение).	Участники проектируют и реализуют в «Берлоге» короткую сцену (1–2 минуты геймплея), где явно показан конфликт. Сцена должна использовать машины состояний (минимум два персонажа с состояниями). Проведение мини-турнира: сравнение работ, голосование за лучшую реализацию конфликта.	Соревнование. Турнир
27.			**ООП в «Берлоге». Введение в ООП. Создание объектов с методами и свойствами**	2	Понятие объектно-ориентированного программирования (объекты, свойства, методы). Как ООП применяется в играх. В «Берлоге» каждый спрайт – это объект, можно задавать его переменные (свойства) и блоки действий (методы).	Создание пользовательского объекта «Игровой персонаж»: добавление свойств «сила», «скорость», «здоровье». Создание метода «атаковать», который уменьшает здоровье цели. Демонстрация вызова метода по событию.	Контроль результатов творческой деятельности (проект с объектом, имеющим свойства и методы)
28.			**Реализация простой системы объектов с параметрами и действиями**	2	Инкапсуляция: объект сам отвечает за свои данные. Обмен сообщениями между объектами. Проектирование системы: например, несколько врагов с разными параметрами (здоровье, урон) и единый метод «получить урон».	Разработка в «Берлоге» системы из 2–3 объектов (игрок, враг, бонус). У каждого объекта – свои параметры (здоровье, сила). Реализация взаимодействия: при касании	Контроль результатов творческой деятельности (рабочая система

						враг атакует игрока, игрок теряет здоровье; при касании бонуса здоровье восстанавливается. Использование методов объектов.	объектов)
29.			**Основы 3D-моделирования. Введение в Blender: интерфейс, навигация, базовые примитивы**	2	Понятие 3D-модели. Области применения (игры, анимация, визуализация). Обзор интерфейса Blender: 3D-окно, панель инструментов, свойства объектов. Навигация (вращение, панорамирование, зуминг). Добавление примитивов (куб, сфера, цилиндр).	Установка Blender (если не установлен). Запуск, знакомство с интерфейсом. Создание сцены с тремя примитивами (куб, сфера, цилиндр), их перемещение, вращение, масштабирование. Сохранение проекта.	Беседа, наблюдение; проверка сохранённой сцены
30.			**Моделирование простых объектов (куб, сфера, цилиндр). Трансформация (перемещение, вращение, масштаб)**	2	Повторение инструментов трансформации (горячие клавиши: G – перемещение, R – вращение, S – масштаб). Применение трансформаций к примитивам для создания простых форм (например, вытянутый куб – кирпич, сплюснутая сфера – блин).	Моделирование трёх простых объектов из примитивов: стол (куб + 4 цилиндра), дерево (цилиндр + сфера), камень (искажённая сфера). Экспорт в формате .obj или .fbx.	Контроль результатов творческой деятельности (файлы моделей)
31.			**Режимы редактирования: вершины, рёбра, грани. Экструдирование**	2	Режим Edit Mode (Tab). Выделение вершин/рёбер/граней. Инструменты: перемещение, вращение, масштабирование выделенного. Экструдирование (E) – создание новой геометрии из выделенной грани. Применение: создание углублений, выступов.	На основе куба создание простого домика (экструдирование крыши, выдавливание окна). Сохранение промежуточных результатов. Практика с инструментом Loop Cut (разрезание) для добавления деталей.	Контроль результатов творческой деятельности (модель домика в формате .blend)

32.			**Создание низкополигональной модели (например, дерево, камень или простой персонаж)**	2	Понятие low-poly (малое количество полигонов), преимущества для игр (производительность, стиль). Техники: использование примитивов, экструдирование без излишней детализации.	По выбору: создание низкополигонального дерева (ствол из цилиндра, крона из нескольких кубов), камня (искажённый куб с экструдированными гранями) или простого персонажа (кубический робот). Соблюдение количества полигонов (не более 500).	Контроль результатов творческой деятельности (готовая low-poly модель)
33.			**Наложение материалов и простых текстур в Blender**	2	Понятие материала (цвет, блеск, прозрачность). Текстура – изображение, накладываемое на модель. Режим Shading Editor: создание материала, изменение базового цвета, добавление шума или изображения. UV-развёртка (базовое понятие).	Присвоение цвета созданной low-poly модели (дерево – коричневый ствол, зелёная крона). Добавление простой текстуры (например, растрового изображения травы) для земли. Настройка отражения (glossy) для металлических деталей.	Контроль результатов творческой деятельности (модель с материалом, скриншот)
34.			**Интеграция 3D-моделей в игровой проект «Берлога» (форматы экспорта, импорт)**	2	Поддерживаемые форматы в «Берлоге» (обычно .obj, .fbx, .dae). Экспорт из Blender: настройки (масштаб, поворот осей). Импорт в «Берлогу»: добавление модели как нового объекта, настройка коллизии, масштабирование.	Экспорт созданной модели в формат .obj. Импорт модели в проект «Берлога» (например, замена стандартного медведя на своего персонажа). Настройка физических свойств (вес, упругость). Тестирование в сцене.	Контроль результатов творческой деятельности (работающая сцена с импортированной моделью)
35.			**Сквозной проект: создание собственной сцены/уровня в «Берлоге» с использованием**	2	Этапы создания уровня: концепт (что это за место), расстановка объектов, настройка логики состояний и событий, тестирование баланса. Требования к проекту: не менее 3 объектов, собранных в Blender, использование	Разработка сцены в командах (или индивидуально). Создание трёх моделей в Blender, импорт, расстановка на сцене. Программирование поведения с использованием состояний и	Наблюдение, промежуточные консультации

			3D-моделей		машин состояний для хотя бы двух персонажей, наличие конфликта (цели).	событий. Тестирование, отладка.	
36.			**Защита проекта (презентация сцены, демонстрация логики состояний и событий). Рефлексия**	2	Структура презентации проекта: название, цель, используемые модели, схема машины состояний, демонстрация геймплея, что получилось, что планируется улучшить.	Публичная защита созданного уровня перед наставником и другими учащимися (7–10 минут). Демонстрация работы состояний, событий, импортированных 3D-моделей. Вопросы от аудитории. Заполнение индивидуального рефлексивного листа (что освоили, что было сложным).	Защита проекта (оценка по критериям: завершённость, работа логики, использование 3D, качество презентации), рефлексия
Модуль 3. Разработка с использованием искусственного интеллекта в среде Godot Engine – 36 часов							
37.			**Введение в Godot Engine. Интерфейс, сцены, узлы (Node)**	2	Обзор Godot Engine: свободный игровой движок, поддержка 2D и 3D, сценарная система. Понятие сцены (Scene) и узла (Node). Иерархия узлов. Основные типы узлов: Node2D, Sprite, CollisionShape, Area2D. Интерфейс редактора (2D/3D/скрипт).	Установка Godot (последней стабильной версии). Создание нового проекта. Добавление узла Sprite, загрузка текстуры-заглушки. Запуск сцены. Сохранение проекта.	Беседа, наблюдение; проверка созданной сцены
38.			**GDScript: переменные, функции, сигналы. Создание первой сцены**	2	Введение в GDScript (синтаксис, отличие от Python). Переменные (var, const), типы (int, float, String, bool). Функции (func), параметры, возвращаемые значения. Сигналы (signal) – механизм событий. Подключение сигнала к функции через редактор или код.	Создание сцены с узлом KinematicBody2D (игрок). Написание скрипта для движения по стрелкам (переменные скорости, функция _process). Подключение сигнала body_entered для обнаружения столкновений. Запуск и тестирование.	Контроль результатов творческой деятельности (работающая сцена с движением)
39.			**ИИ как помощник: обзор**	2	Современные ИИ-инструменты для разработчиков: ChatGPT (веб-	Регистрация/доступ к одному из инструментов (например,	Беседа, демонстрация

			инструментов (ChatGPT, Copilot, Claude). Промпт для генерации кода		интерфейс, API), GitHub Copilot (плагин для IDE), Claude (анализ больших объёмов кода). Что такое промпт (запрос): структура, контекст, примеры. Рекомендации по формулировке: указать язык, версию движка, желаемый функционал, ожидаемый формат вывода.	ChatGPT). Составление промпта для генерации кода движения персонажа в Godot на GDScript. Сравнение сгенерированного кода с кодом из темы 2. Анализ различий.	составленного промпта и полученного кода
40.			**Генерация простых скриптов на GDScript через ИИ. Движение персонажа**	2	Особенности генерации игровой логики: учёт физических свойств (скорость, ускорение), типы движения (клавиши, геймпад). Типичные ошибки ИИ (неправильные имена узлов, устаревший синтаксис). Проверка и адаптация сгенерированного кода.	Запрос к ИИ: «Напиши скрипт для персонажа в Godot 4 (GDScript), который движется влево/вправо, прыгает и проверяет, стоит ли на земле». Интеграция кода в проект. Настройка коллизий, проверка работоспособности.	Контроль результатов творческой деятельности (рабочий скрипт движения и прыжков)
41.			**ИИ для отладки: объяснение ошибок, поиск багов. Рефакторинг с помощью ИИ**	2	Как ИИ может помочь в отладке: скопировать сообщение об ошибке, запросить объяснение. Уточняющие вопросы: «Почему переменная не видна в другой функции?». Рефакторинг: запрос на упрощение кода, улучшение читаемости, выделение методов.	Внесение намеренной ошибки в код движения (например, неверное имя сигнала). Запрос к ИИ с текстом ошибки. Исправление кода по рекомендации ИИ. Запрос на рефакторинг (выделение функции прыжка в отдельный метод). Сравнение исходного и рефакторированного кода.	Контроль результатов творческой деятельности (исправленный и рефакторированный код, отчёт о запросах)
42.			**Генерация игровых объектов с помощью ИИ: шаблоны узлов, экспорт переменных**	2	Создание повторяемых игровых объектов (враги, монеты, бонусы). Использование @export для настройки параметров в редакторе. ИИ для генерации шаблона сцены (PackedScene) и скрипта.	Запрос к ИИ: «Создай скрипт для монеты в Godot 4: при касании игрока проигрывается звук, добавляется 10 очков, монета исчезает». Реализация скрипта. Создание сцены	Контроль результатов творческой деятельности (сцена монеты, работающая

						монеты как отдельного узла, экспорт переменной score_value. Размещение нескольких монет на уровне.	механика сбора)
43.			**ИИ для генерации диалогов и текстовых квестов. Вставка в Godot**	2	Генерация текстового контента с помощью ИИ (NPC-реплики, описания квестов, внутриигровые письма). Структура диалоговой системы в Godot: узлы Control, RichTextLabel, сигналы кнопок. Интеграция сгенерированного текста.	Запрос к ИИ: «Сгенерируй 5 фраз для торговца в фэнтези-игре». Добавление в сцену простого диалогового окна (Panel + RichTextLabel + Button). Вывод случайной фразы при нажатии. Расширение: диалог с выбором ответов.	Контроль результатов творческой деятельности (работающий диалог со сгенерированн ыми текстами)
44.			**Создание интерфейса (UI) в Godot. ИИ-генерация кода для кнопок, панелей, анимаций UI**	2	Элементы UI: Control, Button, Panel, Label, TextureProgressBar. Анимация UI (изменение размера, цвета, прозрачности) с помощью AnimationPlayer. ИИ для генерации кода обработки нажатий и смены сцен.	Запрос к ИИ: «Создай скрипт для главного меню в Godot: кнопки "Старт", "Настройки", "Выход". При нажатии "Старт" загружается сцена уровня». Реализация меню. Добавление анимации появления кнопок (плавное увеличение и изменение цвета).	Контроль результатов творческой деятельности (рабочее главное меню, переход на уровень)
45.			**ИИ для оптимизации кода: анализ сложности, предложения по улучшению**	2	Понятие производительности в играх (FPS, количество объектов, циклы в _process). ИИ как инструмент для оценки сложности алгоритмов и выявления узких мест. Типовые рекомендации: избегать get_node() в каждом кадре, кэшировать ссылки.	Предоставить ИИ фрагмент кода с потенциально неоптимальным циклом (например, поиск всех монет на уровне через get_tree().get_nodes_in_group() в каждом кадре). Запрос «Предложи оптимизацию». Рефакторинг кода по рекомендации. Измерение	Контроль результатов творческой деятельности (оптимизирова нный код, краткий отчёт о замерах)

						производительности до и после (отображение FPS).	
46.			**Генерация описаний предметов, способностей, внутриигровых подсказок через ИИ**	2	Использование ИИ для наполнения игры контентом (лор, описание предметов, названия способностей, подсказки для загрузочных экранов). Форматирование вывода (JSON, CSV) для удобной вставки в Godot.	Запрос к ИИ: «Сгенерируй список из 5 предметов для RPG-игры. Для каждого укажи название, описание, редкость (обычный, редкий, эпический) и эффект (+5 здоровья, +2 силы) в формате JSON». Импорт JSON в Godot (файл items.json). Создание сцены для отображения случайного предмета с использованием данных.	Контроль результатов творческой деятельности (работающее отображение предметов из сгенерированного JSON)
47.			**ИИ как архитектурный советник: проектирование сцен, структуры проекта**	2	Понятие архитектуры проекта: как организованы сцены, глобальные автозагрузки (singletons), сигналы для связи узлов. ИИ может предложить схему взаимодействия компонентов. Пример: «Как организовать систему здоровья игрока, которая обновляет интерфейс и вызывает загрузку сцены поражения?».	Запрос к ИИ описать структуру для простой игры (например, платформер с монетами, врагами и счетом очков). Построение схемы на основе рекомендации (узлы: Player, Enemy, UI, GameManager автозагрузка). Реализация предложенной архитектуры в проекте (хотя бы каркас).	Контроль результатов творческой деятельности (схема архитектуры, реализованный каркас)
48.			**Разработка прототипа простой 2D-игры (платформер или карточная коллекционная стратегия) с помощью ИИ**	2	Что такое прототип (минимальная версия с ключевой механикой). Варианты: платформер (прыжки, сбор предметов, враг) или карточная ККИ (выбор карты, атака, изменение очков здоровья). Роль ИИ на этапе прототипирования: генерация базового кода, ускорение итераций.	Команды выбирают жанр. С помощью ИИ генерируют: для платформера – скрипты движения, прыжков, сбора монет; для ККИ – скрипты выбора карт, подсчёта урона. Интеграция в Godot, получение первого играбельного	Наблюдение, демонстрация прототипа (к концу занятия)

						прототипа (без финальной графики).	
49.			**Итеративная разработка: запрос к ИИ на добавление функционала (напр. прыжки, сбор монет и пр.)**	2	Итеративный процесс: начать с минимума, затем запрашивать у ИИ код для новой функции. Как правильно формулировать запрос в контексте существующего кода (предоставлять фрагменты, указывать имена узлов). Управление сложностью.	На основе прототипа из темы 12 команды формулируют задачу на добавление одной новой функции (например, «добавить врага, который движется влево-вправо» или «добавить анимацию выбора карты»). Запрос к ИИ, интеграция, тестирование. Документирование изменений.	Контроль результатов творческой деятельности
50.			**ИИ для генерации тестовых сценариев и документации (README, комментарии)**	2	Зачем нужны тестовые сценарии (ручное тестирование: "пройти уровень без сбора монет", "столкнуться с врагом трижды"). ИИ может сгенерировать список чек-листов. Документация: комментарии в коде, README для репозитория.	Запрос к ИИ: «Сгенерируй 5 тестовых сценариев для платформера с монетами и врагом». Запрос: «Напиши README для игры [название] на Godot: описание, управление, установка». Добавление комментариев к сложным функциям с помощью ИИ (запрос «прокомментируй этот код»).	Контроль результатов творческой деятельности
51.			**Сравнение кода, написанного самостоятельно и сгенерированного ИИ. Риски и ограничения**	2	Обсуждение различий: стиль, эффективность, избыточность, безопасность. Риски: слепое копирование кода, неподходящие архитектурные решения, генерация неоптимальных или устаревших конструкций. Ограничения ИИ (отсутствие понимания контекста всего проекта).	Рефакторинг кода, сгенерированных ИИ ранее для улучшения или адаптации. Сравнение двух версий (читаемость, количество строк, производительность). Групповое обсуждение.	Беседа, письменные выводы

52.			**Работа над финальным проектом (своя мини-игра на Godot). ИИ как помощник на всех этапах**	2	Этапы разработки финального проекта: концепт, прототип, полировка, тестирование. Роль ИИ на каждом этапе: генерация идей, кода, диалогов, UI, документации, тестов. Самостоятельная организация работы.	Команды (или индивидуальные участники) разрабатывают мини-игру (продолжительность геймплея 2–5 минут) с использованием Godot и ИИ-инструментов. Допускается использование всех изученных приёмов. Наставник проводит промежуточные консультации, помогает сформулировать промпты.	Наблюдение, промежуточные отчёты
53.			**Презентация финального проекта. Демонстрация: какие части сделаны с помощью ИИ**	2	Структура презентации: название, жанр, основные механики, демонстрация геймплея, перечень использованных ИИ-инструментов, конкретные примеры промптов и сгенерированных фрагментов. Оценка вклада ИИ и собственного вклада.	Публичная презентация мини-игры перед наставником и другими учащимися (5–7 минут). Демонстрация работы игры. Показ исходного кода, где видно использование ИИ (комментарии или отдельный слайд). Ответы на вопросы.	Контроль результатов творческой деятельности (оценка игры и презентации)
54.			**Рефлексия: этика использования ИИ, авторство, перспективы. Защита проекта**	2	Этические аспекты: плагиат, авторство сгенерированного кода, указание использования ИИ в документации. Перспективы развития. Защита проекта как итоговая аттестация.	Заполнение рефлексивного листа (что дало использование ИИ, какие были трудности, как изменилось отношение к программированию). Групповая дискуссия «Должен ли разработчик уметь писать код без ИИ?». Финальная защита проекта с выставлением оценки (критерии: завершённость, стабильность, использование ИИ, понимание кода).	Защита проекта, беседа, письмен
Модуль 4. Web-разработка с использованием искусственного интеллекта - 36 часов							

55.			**Введение в веб-разработку. Клиент-сервер. HTML, CSS, JS – базовое повторение. ИИ-помощники написания кода**	2	Архитектура клиент-сервер: роль браузера (клиент) и веб-сервера. Основы HTTP (запрос, ответ, методы GET/POST). Повторение HTML (структура страницы, формы), CSS (цвета, шрифты, позиционирование), JavaScript (переменные, функции, обработка событий). Обзор ИИ-инструментов для веб-разработки (ChatGPT, Copilot, Codeium).	Написание простой HTML-страницы с формой и кнопкой. Добавление CSS-стилей. Написание JS-обработчика нажатия кнопки (вывод сообщения). Запрос к ИИ: «Создай форму логина на HTML/CSS с полями username и password». Анализ сгенерированного кода.	Беседа, наблюдение; проверка созданной страницы
56.			**Основы Flask: маршруты, рендеринг шаблонов, запуск локального сервера**	2	Введение в Flask (микрофреймворк для Python). Установка Flask через pip. Маршруты (декоратор @app.route). Рендеринг HTML-шаблонов с помощью render_template. Запуск сервера (app.run(debug=True)). Структура проекта (папки templates/, static/).	Установка Flask, создание минимального приложения с двумя маршрутами (/ и /about). Создание шаблона index.html с переменной (например, приветствие по имени). Запуск сервера, проверка в браузере.	Контроль результатов творческой деятельности (работающее Flask-приложение)
57.			**Генерация серверного кода на Python с помощью ИИ (ChatGPT/Copilot). Создание простого веб-приложения**	2	Особенности генерации кода для Flask: указание версии Python, описание маршрутов, шаблонов. Как проверять и адаптировать сгенерированный код (синтаксис, имена файлов).	Запрос к ИИ для создания сервера игры. Интеграция кода в проект. Запуск, тестирование. Расширение: добавить статический CSS-файл.	Контроль результатов творческой деятельности (работающее приложение, сгенерированный код адаптирован)
58.			**Передача данных: GET/POST параметры. Формы и обработка данных**	2	Различие GET (параметры в URL) и POST (в теле запроса). Доступ к GET-параметрам через request.args, к POST – через request.form. Обработка отправки формы. Валидация данных (проверка	Создание формы регистрации (имя, email, пароль). Обработчик формы, который проверяет, что поля не пустые, email содержит '@', пароль	Контроль результатов творческой деятельности (работающая

			на сервере. ИИ для написания валидации		на пустоту, формат email и т.д.).	длиннее 6 символов. Запрос к ИИ на генерацию функции валидации. Вывод сообщений об ошибках на той же странице.	форма с валидацией)
59.			**Сессии и состояние пользователя. Генерация логики входа/регистрации через ИИ**	2	Понятие сессии (состояние между HTTP-запросами). Flask-сессии (from flask import session). Хранение данных (например, session['user_id']). Реализация входа (логин), выхода (logout). Защита паролей (хэширование).	Запрос к ИИ: «Реализуй простую систему логина на Flask: хранение пользователей в словаре (временном), хэширование пароля через werkzeug.security, использование сессий». Интеграция кода. Создание страниц логина, дашборда, выхода.	Контроль результатов творческой деятельности (работающая аутентификация)
60.			**Шаблонизатор Jinja2: динамические страницы, циклы, условия. ИИ-генерация шаблонов**	2	Синтаксис Jinja2: переменные {{ }}, операторы {% %}, условия (if), циклы (for). Наследование шаблонов (extends, block). Фильтры (|length, |upper).	Запрос к ИИ: «Создай шаблон Jinja2 для отображения списка товаров: заголовок, таблица с колонками (название, цена, наличие), если список пуст – показать сообщение "Товаров нет"». Интеграция шаблона в Flask-приложение, передача списка из маршрута. Создание базового шаблона base.html с меню.	Контроль результатов творческой деятельности (работающие динамические страницы)
61.			**Работа с базой данных (SQLite через Flask-SQLAlchemy). ИИ для создания моделей и запросов**	2	ORM (SQLAlchemy), модель данных (класс, унаследованный от db.Model). Поля (Integer, String, DateTime). Миграции (Flask-Migrate). Базовые запросы: User.query.filter_by(username=...).first().	Установка Flask-SQLAlchemy. Запрос к ИИ: «Создай модель User для Flask-SQLAlchemy с полями id, username, email, password_hash, created_at. Напиши код для добавления пользователя в базу и выборки	Контроль результатов творческой деятельности (база данных создана, данные

						всех пользователей». Реализация. Вывод списка пользователей на отдельной странице.	добавляются и отображаются)
62.			**Проектирование веб-игры: идея, игровая механика на сервере (например, текстовый квест или кликер). ИИ для генерации описаний**	2	Типы веб-игр с серверной логикой: текстовый квест (выбор действий), кликер (счётчик очков), викторина. Проектирование: какие данные хранить на сервере (очки, прогресс, состояние сессии). ИИ для генерации контента (описания локаций, вопросов).	Команды выбирают тип игры. Запрос к ИИ на генерацию описаний (например, «Сгенерируй 5 локаций для текстового квеста в космосе, укажи описание и возможные действия»). Проектирование структуры базы данных для игры. Создание каркаса (маршруты, шаблоны).	Наблюдение, утверждение идеи и структуры наставником
63.			**Реализация игровой логики на Python (сервер) + простой интерфейс на HTML/CSS. ИИ как помощник в отладке**	2	Перенос игровой механики в код: функции для обработки действий игрока, изменения состояния (счёт, здоровье, инвентарь). Отладка с помощью ИИ – отправка ошибок, запрос исправлений.	Реализация серверной логики выбранной игры (например, для кликера – увеличение счётчика по POST-запросу, для квеста – выбор действия и смена локации). ИИ помогает написать функции. Интерфейс – простой HTML/CSS без JS (обновление страницы при каждом действии). Тестирование, отладка с помощью ИИ.	Контроль результатов творческой деятельности (работающая базовая версия игры)
64.			**Добавление JavaScript на фронтенд: динамические обновления без перезагрузки (fetch**	2	Принципы AJAX. Fetch API: отправка запросов к серверу из JS, обработка ответа (JSON). Обновление DOM без перезагрузки страницы. Сравнение с классическим подходом (перезагрузка).	Формулировка запроса к AI, предполагающая написание последовательности операций для отправки данных и обновления отображаемого состояния. Встраивание в	Контроль результатов творческой деятельности (динамическая игра без

			API). ИИ для генерации fetch-запросов			интерактивное приложение, где воздействие инициирует передачу сведений, а в ответ происходит модификация внутреннего состояния и возврат формализованного сообщения. Вариативное применение для сценариев с последовательным выбором вариантов и динамическим изменением контекста.	перезагрузки)
65.			**Стилизация и адаптив: CSS + ИИ. Анимации интерфейса**	2	Адаптивный дизайн (медиа-запросы, гибкие сетки). CSS-анимации (transition, keyframes). ИИ для генерации стилей и анимаций (промпт: «Создай адаптивный дизайн для игрового интерфейса: кнопки, панель счёта, карточки локаций»).	Запрос к ИИ на генерацию CSS для улучшения внешнего вида игры. Добавление медиа-запросов для мобильных устройств. Реализация анимации наведения кнопок, появления очков. Интеграция в проект.	Контроль результатов творческой деятельности (стилизованная адаптивная игра)
66.			**Подготовка к публикации: переменные окружения, requirements.txt, статические файлы. ИИ для генерации инструкций**	2	Переменные окружения (секретный ключ, настройки БД). Файл .env, библиотека python-dotenv. Создание requirements.txt (заморозка зависимостей). Структура статических файлов (CSS, JS, изображения).	Запрос к ИИ: «Создай инструкцию по подготовке Flask-приложения к деплою: список переменных окружения, команды для создания requirements.txt, настройка статики». Выполнение инструкции: создание .env, генерация requirements.txt (pip freeze), проверка, что статические файлы подгружаются правильно.	Контроль результатов творческой деятельности (файл requirements.txt, корректная работа приложения с переменными окружения)
67.			**Публикация на**	2	Обзор бесплатных хостингов для Flask	Регистрация на	Контроль

			PythonAnywhere (или аналогичном хостинге). Деплой веб-игры. ИИ для решения проблем при публикации		(PythonAnywhere, Render, Railway). Этапы деплоя: загрузка кода, настройка виртуального окружения, указание WSGI-файла, настройка переменных окружения. Возможные проблемы (пути к файлам, версии Python).	PythonAnywhere. Загрузка проекта через Git или вручную. Настройка веб-приложения. Запуск игры в интернете. При возникновении ошибок – запрос к ИИ (текст ошибки), получение рекомендаций.	результатов творческой деятельности (игра доступна по публичной ссылке)
68.			**Тестирование опубликованной игры. Сбор обратной связи. Итеративные улучшения с помощью ИИ**	2	Методы тестирования веб-приложения (функциональное, кроссбраузерное, нагрузочное). Сбор обратной связи от одногруппников (опрос, комментарии). Приоритизация улучшений. ИИ для генерации идей по улучшению.	Учащиеся играют в проекты друг друга, выявляют баги и дают предложения. Каждая команда собирает обратную связь (например, в Trello или Google Forms). Запрос к ИИ: «Предложи 3 улучшения для моей веб-игры [описание]». Реализация одного-двух улучшений (итерация).	Наблюдение, отчёт о внесённых улучшениях
69.			**Защита проекта: демонстрация работающей онлайн-игры, презентация использования ИИ на всех этапах. Рефлексия**	2	Структура защиты: URL игры, описание механик, демонстрация (2–3 минуты геймплея), рассказ о том, как использовался ИИ (генерация кода, отладка, контент, стили), рефлексия (что далось легко, что трудно).	Публичная защита проектов (по 7–10 минут на команду). Демонстрация живой онлайн-игры. Ответы на вопросы. Заполнение рефлексивного листа.	Защита проекта (оценка наставником и взаимная оценка)
70.			**Обобщение тем курса. Обсуждение вектора дальнейшего развития**	2	Систематизация знаний, полученных во втором году (командная разработка, событийное программирование в «Берлоге», 3D-моделирование, Godot + ИИ, веб-разработка с ИИ). Связь между модулями. Возможные пути продолжения: участие в хакатонах,	Групповая дискуссия «Какой модуль был самым полезным и почему?». Составление личного плана на следующие 6 месяцев (курсы, проекты, чтение).	Беседа, наблюдение

					углублённое изучение движков, фулстек-разработка, DevOps.		
71.			**Подготовка к итоговой выставке**	2	Требования к выставочному стенду: ноутбук/планшет с демо, плакат (описание проекта), QR-код на игру (для веб-проектов). Краткая речь (1–2 минуты). Рекомендации по визуальному оформлению.	Команды готовят выставочные материалы: создают постер (Canva или PowerPoint), продумывают интерактивную демонстрацию, репетируют презентацию. Проведение внутреннего прогона.	Контроль результатов творческой деятельности (готовые стендовые материалы)
72.			**Проведение итоговой выставки проектов**	2	Организация выставки (зоны, тайминг, жюри). Критерии оценки: техническая сложность, дизайн, работа ИИ, презентационные навыки. Правила голосования.	Участвуют все модульные проекты (каждый выбирает один или несколько). Презентация посетителям (педагогам, родителям, другим учащимся). Ответы на вопросы. Подведение итогов, награждение.	Защита проекта (оценка внешним жюри, приз зрительских симпатий)
			ИТОГО	144			

Приложение №2
к дополнительной
общеразвивающей программе
«IT-КВАНТУМ. ВВЕДЕНИЕ В
КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ И НАУКИ
КАБАРДИНО-БАЛКАРСКОЙ РЕСПУБЛИКИ
ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБЩЕОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
«ДЕТСКАЯ АКАДЕМИЯ ТВОРЧЕСТВА «СОЛНЕЧНЫЙ ГОРОД»

РАБОЧАЯ ПРОГРАММА ВОСПИТАНИЯ ОБУЧАЮЩИХСЯ НА 2026-2027 УЧЕБНЫЙ ГОД К ДОПОЛНИТЕЛЬНОЙ ОБЩЕРАЗВИВАЮЩЕЙ ПРОГРАММЕ «IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»

Уровень программы: базовый
Год обучения: первый, второй
Номер группы:
Адресат: 11-15 лет
Автор-составитель: Кравченко Алексей Михайлович,
педагог дополнительного образования

Нальчик, 2026

СОДЕРЖАНИЕ

1. Особенности воспитательного процесса организуемого в ЦДОД

Воспитательный процесс в Центре дополнительного образования детей ГБОУ «ДАТ «Солнечный город» Минпросвещения КБР (далее по тексту ЦДОД, Центр) по дополнительной общеразвивающей программе «IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»организован на основе настоящей рабочей программы воспитания, сформированной на период 2026 - 2027 гг., и направлен на развитие личности, создание условий для самоопределения и социализации обучающихся на основе социокультурных, духовно-нравственных ценностей и принятых в российском обществе правил и норм поведения в интересах человека, семьи, общества и государства, формирование у обучающихся чувства патриотизма, гражданственности, уважения к памяти защитников Отечества и подвигам Героев Отечества, закону и правопорядку, человеку труда и старшему поколению, взаимного уважения, бережного отношения к культурному наследию и традициям многонационального народа Российской Федерации, природе и окружающей среде.

Воспитательный процесс в Центре дополнительного образования детей ГБОУ «ДАТ «Солнечный город» Минпросвещения КБР основан на следующих традициях воспитания:

– гуманистический характер воспитания и обучения;

– приоритет общечеловеческих ценностей, жизни и здоровья человека, свободного развития личности;

– воспитание гражданственности, трудолюбия, уважения к правам и свободам человека, любви к окружающему миру, Родине, семье;

– развитие национальных и региональных культурных традиций в условиях многонационального государства;

– демократический государственно-общественный характер управления образованием.

Основными традициями воспитания в Центре дополнительного образования детей ГБОУ «ДАТ «Солнечный город» Минпросвещения КБР являются следующие:

- обеспечение комфортной эмоциональной среды взаимодействия всех участников образовательного процесса, создание ситуации успеха образования;

- содействие формированию личности обучающихся, развитию творческих способностей обучающихся в условиях инновационной развивающейся образовательной среды, создание позитивной мотивации к обучению,

- воспитание гармонично развитой и социально ответственной личности гражданина и патриота, на основе истории и традиций России и Кабардино-Балкарской Республики;

- формирование здорового образа жизни, успешной социальной адаптации.

2. Цель и задачи воспитания

Современный национальный воспитательный идеал — это высоконравственный, творческий, компетентный гражданин России, принимающий судьбу Отечества как свою личную, осознающий ответственность за настоящее и будущее своей страны, укоренённый в духовных и культурных традициях многонационального народа Российской Федерации.

Основная цель воспитания – личностное развитие обучающихся - в ЦДОД основывается на базовых для нашего общества ценностях, таких как, семья, труд, отечество, природа, мир, знания, культура, здоровье, человек, и проявляется:

1) в усвоении ими знаний основных норм, которые общество выработало на основе этих ценностей (то есть, в усвоении ими социально значимых знаний);

2) в развитии у них позитивного отношения к этим общественным ценностям (то есть в развитии у них социально-значимых отношений);

3) в приобретении ими соответствующего этим ценностям опыта поведения, опыта применения сформированных знаний и отношений на практике (то есть в приобретении ими опыта осуществления социально-значимой деятельности, в том числе профессионально ориентированной).

Данная цель ориентирует педагогов ЦДОД на обеспечение позитивной динамики развития личности обучающихся.

Достижению поставленной цели воспитания обучающихся будет способствовать решение следующих основных задач:

– освоение обучающимися ценностно-нормативного и деятельностно-практического аспекта отношений человека с человеком, патриота с Родиной, гражданина с правовым государством и гражданским обществом, человека с природой, с искусством и т.д.;

– вовлечение обучающихся в процессы самопознания, самопонимания, содействие обучающимся в соотнесении представлений о собственных возможностях, интересах, ограничениях с запросами и требованиями окружающих людей, общества, государства;

– помощь в личностном самоопределении, проектировании индивидуальных образовательных траекторий и образа будущей профессиональной деятельности, поддержка деятельности обучающихся по саморазвитию;

– овладение обучающимися социальными, регулятивными и коммуникативными компетенциями, обеспечивающими ему индивидуальную успешность в общении с окружающими, результативность в социальных практиках, в процессе сотрудничества со сверстниками, старшими и младшими.

3. Характеристика объединения «IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»

Деятельность объединения «IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ» имеет техническую направленность.

Количество обучающихся объединения «IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ» составляет человек.

Из них мальчиков – ___, девочек – ___.

Обучающиеся имеют возрастную категорию детей от 10 до 16 лет.

Формы работы – индивидуальные и групповые.

4. Виды, формы и содержание воспитательной деятельности

Работа с обучающимися

Практическая реализация цели и задач воспитания осуществляется в рамках следующих направлений воспитательной работы ЦДОД:

1) становление личности в духе патриотизма и гражданственности;
2) социализация и духовно-нравственное развитие личности;
3) бережное отношение к живой природе, культурному наследию и народным традициям;
4) воспитание у обучающихся уважения к труду и людям труда, трудовым достижениям; профессиональная ориентация;
5) воспитание познавательных интересов обучающихся: потребность в приобретении новых знаний, интереса к творческой деятельности;
6) физическое воспитание, содействие здоровому образу жизни;
7) развитие социального партнерства в воспитательной деятельности ЦДОД;
8) конструирования содержания образования как проектов индивидуальной и совместной деятельности преподавателей и участников по достижению образовательных результатов и завершается их защитой.

Перечисленные направления воспитательной работы представлены в соответствующих модулях.

Работа с родителями

Работа с родителями в рамках воспитательной деятельности объединения осуществляется по следующим направлениям:

1) организация системы индивидуальной и коллективной работы сродителями (тематические беседы, собрания, индивидуальные консультации);

2) содействие сплочению родительского коллектива и вовлечение в жизнедеятельность творческого объединения (организация и проведение открытых занятий и иных мероприятий с участием родителей в течение года);

3) оформление информационных уголков для родителей по вопросам воспитания детей.

Реализация конкретных форм и методов воспитательной работы представлены в календарном плане воспитательной работы, утверждаемом ежегодно на предстоящий учебный год, на основе направлений воспитательной работы, установленных в настоящей рабочей программе воспитания.

4.1. Модуль «Гражданин и патриот»

Цель модуля: развитие личности обучающегося на основе формирования у обучающихся чувства патриотизма, гражданственности, уважения к памяти защитников Отечества и подвигам Героев Отечества, закону и правопорядку.

Задачи модуля:

– формирование знаний обучающихся о символике России;

– воспитание у обучающихся готовности к выполнению гражданского долга и конституционных обязанностей по защите Родины;

– формирование у обучающихся патриотического сознания, чувства верности своему Отечеству;

– развитие у обучающихся уважения к памяти защитников Отечества и подвигам Героев Отечества, историческим символам и памятникам Отечества;

– формирование российской гражданской идентичности, гражданской позиции активного и ответственного члена российского общества, осознающего свои конституционные права и обязанности, уважающего закон и правопорядок, обладающего чувством собственного достоинства, осознанно принимающего традиционные национальные и общечеловеческие гуманистические и демократические ценности;

– развитие правовой и политической культуры обучающихся, расширение конструктивного участия в принятии решений, затрагивающих их права и интересы, в том числе в различных формах общественной самоорганизации, самоуправления, общественно значимой деятельности; развитие в молодежной среде ответственности, принципов коллективизма и социальной солидарности;

– формирование приверженности идеям интернационализма, дружбы, равенства, взаимопомощи народов; воспитание уважительного отношения к национальному достоинству людей, их чувствам, религиозным убеждениям;

– формирование установок личности, позволяющих противостоять идеологии экстремизма, национализма, ксенофобии, коррупции, дискриминации по социальным, религиозным, расовым, национальным признакам и другим негативным социальным явлениям;

– формирование антикоррупционного мировоззрения.

Формы реализации модуля:

№ п/п	Наименование мероприятия	Срок выполнения	Ответственный исполнитель	Планируемый результат
1	Серия занятий посвященных Дню Народного единства	ноябрь	Кравченко А.М.	формирование у обучающихся патриотического сознания
2	Серия занятий посвященных Дню	май	Кравченко А.М.	формирование у обучающихся

	Победы в Великой Отечественной войне.			патриотического сознания

4.2. Модуль «Социализация и духовно-нравственное развитие»

Цель модуля: создание условий для самоопределения и социализации обучающихся на основе социокультурных, духовно-нравственных ценностей и принятых в российском обществе правил и норм поведения в интересах человека, семьи, общества и государства, формирование у обучающихся уважения к старшему поколению.

Задачи модуля:

– воспитание здоровой, счастливой, свободной личности, формирование способности ставить цели и строить жизненные планы;

– реализация обучающимися практик саморазвития и самовоспитания в соответствии с общечеловеческими ценностями и идеалами гражданского общества;

– формирование позитивных жизненных ориентиров и планов;

– формирование у обучающихся готовности и способности к образованию, в том числе самообразованию, на протяжении всей жизни; сознательное отношение к непрерывному образованию как условию успешной профессиональной и общественной деятельности;

– формирование у обучающихся ответственного отношения к своему здоровью и потребности в здоровом образе жизни, физическом самосовершенствовании, занятиях спортивно-оздоровительной деятельностью, развитие культуры безопасной жизнедеятельности, профилактику наркотической и алкогольной зависимости, табакокурения и других вредных привычек;

– формирование бережного, ответственного и компетентного отношения к физическому и психологическому здоровью – как собственному, так и других людей, умение оказывать первую помощь, развитие культуры здорового питания;

– развитие способностей к сопереживанию и формированию позитивного отношения к людям, в том числе к лицам с ограниченными возможностями здоровья и людям с инвалидностью;

– формирование выраженной в поведении нравственной позиции, в том числе способности к сознательному выбору добра, нравственного сознания и поведения на основе усвоения общечеловеческих ценностей и нравственных чувств (чести, долга, справедливости, милосердия и дружелюбия);

– развитие компетенций сотрудничества со сверстниками, детьми младшего возраста, взрослыми в образовательной, общественно полезной, учебно-исследовательской, проектной и других видах деятельности;

– развитие культуры межнационального общения;

– развитие в молодежной среде ответственности, принципов коллективизма и социальной солидарности;

– формирование уважительного отношения к родителям и старшему поколению в целом, готовности понять их позицию, принять их заботу, готовности договариваться с родителями и членами семьи в решении вопросов ведения домашнего хозяйства, распределения семейных обязанностей;

– воспитание ответственного отношения к созданию и сохранению семьи на основе осознанного принятия ценностей семейной жизни;

– содействие в осознанной выработке собственной позиции по отношению к общественно-политическим событиям прошлого и настоящего на основе осознания и осмысления истории, духовных ценностей и достижений нашей страны;

– формирование толерантного сознания и поведения в поликультурном мире, готовности и способности вести диалог с другими людьми, достигать в нем взаимопонимания, находить общие цели и сотрудничать для их достижения.

Формы реализации модуля:

№ п/п	Наименование мероприятия	Срок выполнения	Ответственный исполнитель	Планируемый результат
1	Занятие посвященное основам кибер-безопасности.	В течение года	Кравченко А.М.	реализация обучающимися практик саморазвития и самовоспитания в соответствии с общечеловеческими ценностями и идеалами гражданского общества; формирование позитивных жизненных ориентиров и планов; формирование у обучающихся готовности и способности к образованию, в том числе самообразованию, на протяжении всей жизни; сознательное отношение к непрерывному образованию как условию успешной профессиональной и общественной деятельности;

4.3. Модуль «Окружающий мир: живая природа, культурное наследие и народные традиции»

Цель модуля: формирование у обучающихся чувства бережного отношения к живой природе и окружающей среде, культурному наследию и традициям многонационального народа России.

Задачи модуля:

– формирование у обучающихся готовности и способности к самостоятельной, творческой и ответственной деятельности;

– развитие у обучающихся экологической культуры, бережного отношения к родной земле, природным богатствам России и мира, понимание влияния социально-экономических процессов на состояние природной и социальной среды;

– воспитание чувства ответственности за состояние природных ресурсов, формирование умений и навыков разумного природопользования, нетерпимого отношения к действиям, приносящим вред экологии; приобретение опыта эколого-направленной деятельности;

– воспитание эстетического отношения к миру, включая эстетику быта, научного и технического творчества, спорта, общественных отношений;

– формирование способности к духовному развитию, реализации творческого потенциала в учебной, профессиональной деятельности на основе нравственных установок и моральных норм, непрерывного образования, самовоспитания и универсальной духовно-нравственной компетенции – «становиться лучше»;

– формирование мировоззрения, соответствующего современному уровню развития науки и общественной практики, основанного на диалоге культур, а также на признании

различных форм общественного сознания, предполагающего осознание своего места в поликультурном мире;

– формирование чувства любви к Родине на основе изучения культурного наследия и традиций многонационального народа России.

Формы реализации модуля:

№ п/п	Наименование мероприятия	Срок выполнения	Ответственный исполнитель	Планируемый результат
1	Занятие Экологического воспитания	По графику	Кравченко А.М.	формирование мировоззрения, соответствующего современному уровню развития науки и общественной практики, основанного на диалоге культур, а также на признании различных форм общественного сознания, предполагающего осознание своего места в поликультурном мире;
2	«Культура народов КБР»	май	Кравченко А.М.	Повышение интереса к истории, культуре, традициям народов КБР

4.4. Модуль «Профориентация»

Цель модуля: создание условий для удовлетворения потребностей обучающихся в интеллектуальном, культурном и нравственном развитии в сфере трудовых и социально-экономических отношений посредством профессионального самоопределения.

Задачи модуля:

– развитие общественной активности обучающихся, воспитание в них сознательного отношения к труду и народному достоянию;

– формирование у обучающихся потребности трудиться, добросовестно, ответственно и творчески относиться к разным видам трудовой деятельности;

– формирование soft-skills-навыков и профессиональных компетенций;

– формирование осознания профессиональной идентичности (осознание своей принадлежности к определённой профессии и профессиональному сообществу);

– формирование чувства социально-профессиональной ответственности, усвоение профессионально-этических норм;

– осознанный выбор будущего профессионального развития и возможностей реализации собственных жизненных планов;

– формирование отношения к профессиональной деятельности как возможности участия в решении личных, общественных, государственных, общенациональных проблем.

Формы реализации модуля:

№ п/п	Наименование мероприятия	Срок выполнения	Ответственный исполнитель	Планируемый результат
1	Тестирование «Какая IT специальность тебе подходит»	в течение года	Кравченко А.М.	осознанный выбор будущего профессионального развития и возможностей

				реализации собственных жизненных планов

4.5. Модуль «Социальное партнерство в воспитательной деятельности ЦДОД»

Цель модуля: усиление взаимодействия ЦДОД с организациями, созданными по инициативе обучающихся, с общественными движениями, органами власти и другими образовательными организациями.

Задачи модуля:

– расширение пространства социального партнерства, развитие различных форм взаимодействия его субъектов в сфере воспитательной деятельности;

– распространение опыта и совместное проведение конференций, семинаров и других учебно-воспитательных мероприятий;

– развитие сотрудничества с социальными партнёрами с целью повышения психолого-педагогического мастерства, уровня культуры педагогических работников ЦДОД;

– организация сотрудничества ЦДОД с правоохранительными органами по предупреждению правонарушений среди обучающихся;

– поддержка и продвижение социально значимых инициатив обучающихся и (или) их организаций/ объединений в ЦДОД, городе, республике;

– формирование корпоративной культуры ЦДОД (принадлежности к единому коллективу, формирование традиций, корпоративной этики);

– создание в ЦДОД музеев, историко-патриотических клубов, литературно-творческих объединений, научных обществ с привлечением ветеранов труда, деятелей науки, культуры и искусства;

– создание положительного имиджа ЦДОД, продвижение на уровне города, республики.

Формы реализации модуля:

№ п/п	Наименование мероприятия	Срок выполнения	Ответственный исполнитель	Планируемый результат
1	Публикация в социальных медиа информации о проводимых выставках и конкурсах	в течение года	Кравченко А.М.	создание положительного имиджа ЦДОД, продвижение на уровне города, республики.

4.6. Модуль «Работа с родителями»

Цель модуля: формирование партнерских отношений между педагогами с родителями (законными представителями) для создания благоприятной, развивающей среды, способствующей самореализации ребенка.

Задачи:

- повышение педагогической культуры родителей, обучение методам поддержки творческого и личностного развития ребенка.
- активное включение родителей в образовательный и воспитательный процесс (совместные проекты, праздники, открытые занятия).
- консультирование родителей по вопросам возрастных особенностей, взаимоотношений и

коррекции поведения.

- укрепление сотрудничества, повышение качества образовательных услуг на основе учета запросов семьи.
- содействие формированию здорового образа жизни в семье и профилактика асоциального поведения.

Формы реализации модуля:

№ п/п	Наименование мероприятия	Срок выполнения	Ответственный исполнитель	Планируемый результат
1.	«Партнеры в воспитании-педагоги и семья».	ноябрь	Кравченко А.М.	Повышение педагогической культуры родителей — ключевой аспект такого взаимодействия, который способствует совершенствованию семейного воспитания, гармонизации детско-родительских отношений и повышению эффективности воспитательного процесса.
2.	«Корни моей семьи».	декабрь	Кравченко А.М.	Изучение корней семьи помогает лучше понять свою историю, укрепить семейные связи, сохранить культурное наследие для будущих поколений и развить интерес к истории страны.
3.	«Калейдоскоп профессий»	март	Кравченко А.М.	Склонности и интересы детей в выборе профессии, осознание своих способностей и ценностей.

4.7. Модуль «IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»

Цель модуля: Ранняя профессиональная ориентация обучающихся, повышение престижа инженерных и научных профессий, подготовка кадрового резерва для глобального технологического лидерства России в области информационных технологий.

Задачи модуля:

- научить детей обсуждать различные вопросы, работать с различными источниками информации.
- воспитать чувство коллективизма, взаимопомощи и взаимовыручки.
- ранняя профессиональная ориентация обучающихся
- воспитать дисциплинированность и ответственность за свои действия и свой проект.

Формы реализации модуля:

№ п/п	Наименование мероприятия	Срок выполнения	Ответственный исполнитель	Планируемый результат
1	Участие в Национальной	По графику	Кравченко А.М.	повышение престижа инженерных и научных

	Технологической олимпиаде (НТО)			профессий, осознанный выбор будущего профессионального развития и возможностей реализации собственных жизненных планов

5. Основные направления самоанализа воспитательной работы

Самоанализ организуемой в ЦДОД воспитательной работы осуществляется по направлениям воспитательной работы и проводится с целью выявления основных проблем воспитания обучающихся и последующего их решения.

Самоанализ осуществляется ежегодно силами Центра дополнительного образования детей.

Основными принципами, на основе которых осуществляется самоанализ воспитательной работы в ЦДОД, являются:

- принцип гуманистической направленности осуществляемого анализа;
- принцип приоритета анализа сущностных сторон воспитания: изучение содержания и разнообразия деятельности, характер общения и отношений между обучающимися и педагогическими работниками ЦДОД;
- принцип развивающего характера осуществляемого анализа: грамотная постановка педагогическими работниками ЦДОД цели и задач воспитания, умелого планирования воспитательной работы, адекватного подбора видов, форм и содержания совместной деятельности с обучающимися;
- принцип разделенной ответственности за результаты личностного развития обучающихся: личностное развитие обучающихся – это результат как социального воспитания (в котором образовательная организация участвует наряду с другими социальными институтами), так и стихийной социализации, и саморазвития обучающихся.

Основными направлениями анализа, организуемого в ЦДОД воспитательного процесса, являются:

- результаты воспитания, социализации и саморазвития обучающихся;
- состояние организуемой в ЦДОД совместной деятельности обучающихся и педагогических работников.

Направления анализа воспитательного процесса	**Критерий анализа**	**Способ получения информации о результатах воспитания**	**Результат анализа**
Результаты воспитания, социализации и саморазвития обучающихся	Динамика личностного развития обучающихся	Педагогическое наблюдение	Получение представления о том, какие прежде существовавшие проблемы личностного развития обучающихся удалось решить за прошедший учебный год; какие проблемы решить не удалось и почему; какие новые проблемы появились, над чем далее предстоит работать педагогическим работникам ЦДОД
Состояние организуемой в ЦДОД совместной деятельности	Наличие в ЦДОД интересной, событийно насыщенной	Беседы с обучающимися, педагогическими работниками ЦДОД, при	Получение представления о качестве совместной деятельности обучающихся и педагогических работников ЦДОД по направлениям:

обучающихся и педагогических работников и	и личностно развивающей совместной деятельности обучающихся и педагогических работников	необходимости – их анкетирование	– патриотизм и гражданственность; – социализация и духовно-нравственное развитие; – окружающий мир: живая природа, культурное наследие и народные традиции; – профориентация; – социальное партнерство в воспитательной деятельности образовательной организации; – профессиональное воспитание_____

Анализ организуемого в ЦДОД воспитательного процесса осуществляется заместителем руководителя образовательной организации по учебно-воспитательной работе (совместно с членами комиссии, при необходимости) с последующим обсуждением его результатов на заседании отдела социально-гуманитарной направленности ЦДОД и на Педагогическом совете.

Итогом самоанализа воспитательной работы является перечень выявленных проблем:

потребность в проектировании новой модели личности детей, построенной на профессионально-личностных компетенциях;

низкий уровень общей культуры обучающихся;

недостаточно высокая активность педагогических работников и обучающихся в конкурсном движении.

Список использованной литературы:

Нормативно-правовые документы:

1. Федеральный закон Российской Федерации от 29.12.2012г. № 273-ФЗ «Об образовании в Российской Федерации».
2. Федеральный закон от 31 июля 2020 г. № 304-ФЗ “О внесении изменений в Федеральный закон «Об образовании в Российской Федерации» по вопросам воспитания обучающихся”.
3. Концепция развития дополнительного образования детей, утвержденная распоряжением Правительства Российской Федерации от 04.09.2014г. № 1726-р.
4. Приказ Минобрнауки РФ от 27.07.2022г. №629 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам».

Литература:

1. Письмо Минпросвещения КБР от 20.06.2024г. №22-16-17/5456 «О направлении методических рекомендаций» (вместе с «Методическими рекомендациями по разработке и реализации дополнительных общеразвивающих программ (включая разноуровневые и модульные), «Методическими рекомендациями по разработке и экспертизе качества авторских дополнительных общеразвивающих программ»).
2. «Примерная программа воспитания для образовательных организаций общего образования». /Институт стратегии развития образования РАО, утверждена на заседании Федерального учебно-методического объединения по общему образованию 2 июня 2020 г.
3. Воспитание. Авторские программы школ России (избранные модули): Сборник /Составители Н.Л. Селиванова, П.В. Степанов, В.В. Круглов, И.С. Парфенова, И.В. Степанова, Е.О. Черкашин, И.Ю.Шустова. –М.: ФГБНУ «Институт стратегии развития образования Российской академии образования», 2020.
4. Воспитательный процесс: изучение эффективности: методические рекомендации/ под редакцией Е.Н. Степанова – М., 2011.
5. Кутеева О. Планирование воспитательной работы на основе личностно-ориентированного обучения/ О.Кутеева// Классный руководитель. – 2001. - №1.
6. Каргина З.А. Практическое пособие для педагога дополнительного образования. – Изд. доп. – М.: Школьная Пресса, 2008.
7. Маленкова П.И. Теория и методика воспитания/П.И.Маленкова. - М., 2012.
8. Сластенин В.А. Методика воспитательной работы/ В.А. Сластенин. - изд.2-е.-М., 2014.

Интернет-источники:

https://pandia.ru/text/77/456/934.php - особенности воспитательной работы в системе дополнительного образования;

https://videouroki.net/razrabotki/rabochaya-programma-po-vospitatelnoy-rabote.html - рабочая программа по воспитательной работе;

https://infourok.ru/rabochaya-oprgramma-vospitatelnoy-raboti-328614.html- рабочая программа воспитательной работы.

Календарный план воспитательной работы
объединения «IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»
2026-2027 учебного года

№ п/п	Направление воспитательной работы	Наименование мероприятия	Срок выполнения	Ответственный исполнитель	Планируемый результат
1.	Гражданин и патриот	Серия занятий посвященных Дню Народного единства	ноябрь	Кравченко А.М.	формирование у обучающихся патриотического сознания
		Серия занятий посвященных Дню Победы в Великой Отечественной войне.	май	Кравченко А.М.	формирование у обучающихся патриотического сознания
2.	Социализация и духовно-нравственное развитие	Занятие, посвященное основам кибер-безопасности.	в течение года	Кравченко А.М.	реализация обучающимися практик саморазвития и самовоспитания в соответствии с общечеловеческими ценностями и идеалами гражданского общества; формирование позитивных жизненных ориентиров и планов; формирование у обучающихся готовности и способности к образованию, в том числе самообразованию, на протяжении всей жизни; сознательное отношение к непрерывному образованию как условию успешной профессиональной и общественной деятельности;
3.	Окружающий мир: живая природа, культурное наследие и народные традиции	Занятие экологического воспитания	в течение года	Кравченко А.М.	формирование мировоззрения, соответствующего современному уровню развития науки и общественной практики, основанного на диалоге культур, а также на признании различных форм общественного сознания, предполагающего осознание своего места в поликультурном мире;
		«Культура народов КБР»	май	Кравченко А.М.	Повышение интереса к истории, культуре,

					традициям народов КБР
4.	Профориентация	Тестирование «Какая IT специальность тебе подходит»	в течение года	Кравченко А.М.	осознанный выбор будущего профессионального развития и возможностей реализации собственных жизненных планов
5.	Социальное партнерство в воспитательной деятельности образовательной организации	Публикация в социальных медиа информации о проводимых выставках и конкурсах	в течение года	Кравченко А.М.	создание положительного имиджа ЦДОД, продвижение на уровне города, республики.
6.	Работа с родителями	«Партнеры в воспитании-педагоги и семья».	ноябрь	Кравченко А.М.	Повышение педагогической культуры родителей — ключевой аспект такого взаимодействия, который способствует совершенствованию семейного воспитания, гармонизации детско-родительских отношений и повышению эффективности воспитательного процесса.
		«Корни моей семьи».	декабрь	Кравченко А.М.	Изучение корней семьи помогает лучше понять свою историю, укрепить семейные связи, сохранить культурное наследие для будущих поколений и развить интерес к истории страны.
		«Калейдоскоп профессий»	март	Кравченко А.М.	Склонности и интересы детей в выборе профессии, осознание своих способностей и ценностей.
7.	Модуль «IT-КВАНТУМ. ВВЕДЕНИЕ В КОМПЬЮТЕРНУЮ РАЗРАБОТКУ»	Участие в Национальной Технологической олимпиаде (НТО)	По графику	Кравченко А.М.	повышение престижа инженерных и научных профессий, осознанный выбор будущего профессионального развития и возможностей реализации собственных жизненных планов